VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Servisní program pro kalibraci detektorů plynů

Service Program for Gas Detector Calibration

2016 Lukáš Stehlík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Lukáš Stehlík**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Servisní program pro kalibraci detektorů plynů
Service Program for Gas Detector Calibration

Jazyk vypracování: čeština

Zásady pro vypracování:

Navrhněte a realizujte program, který bude v servisním středisku využíván pro kalibraci detektorů plynů. Pro komunikaci s měřícím zařízením použijte protokol MODBUS RTU a MODBUS ASCII na sériové lince RS485. Navržený program musí být realizován v jazyce C# a musí být funkční na všech verzích OS Windows od verze XP. Funkcionalitu programu ověřte také v OS Linux s využitím frameworku MONO.

1. Popište stávající stav řešení, postup měření, používané přístroje a nedostatky stávajícího řešení. Stanovte požadavky pro nové programové vybavení.
2. Popište způsob komunikace s přístrojovým vybavením a postup měření. Implementujte program pro komunikaci se všemi potřebnými měřícími přístroji.
3. Navrhněte servisní program pro řízení kalibrace detektorů plynů. Při návrhu zohledněte všechny požadavky na proces kalibrace. Návrh aplikace musí být připraven pro více jazykových variant uživatelského rozhraní.
4. Implementujte navržený program.
5. Otestujte navržený program, jeho stabilitu, spolehlivost. Ověřte komunikaci se všemi požadovanými detektory plynů ve všech potřebných pracovních režimech.
6. Proveďte kalibrace detektorů plynů, vyhodnoťte dosažené výsledky, jejich přesnost a spolehlivost.

Seznam doporučené odborné literatury:

[1] Přehled protokolu MODBUS: http://home.zcu.cz/~ronesova/bastl/files/modbus.pdf
[2] MODBUS Protocol Specification: www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf
[3] MODBUS Over Serial Line FOR LEGACY APPLICATIONS ONLY: www.modbus.org/docs/PI_MBUS_300.pdf
[4] Uživatelské příručky SC-CH4, SC-LCD, SC-TOX, SC-TOX-FS, SC-IR, SC-HOUK, ZAM-SERVIS s.r.o.
[5] ČSN EN 50104: Elektrická zařízení pro detekci a měření kyslíku - Požadavky na provedení a metody zkoušek
[6] ČSN EN 45544-1: Ovzduší na pracovišti – El. přístroje používané pro přímou detekci a přímé měření koncentrace toxických plynů a par – Část 1: Všeobecné požadavky a zkušební metody
[7] ČSN EN 45544-2: Část 2: Funkční požadavky na přístroje používané pro měření koncentrací v oblasti limitních hodnot
[8] ČSN EN 45544-3: Část 3: Funkční požadavky na přístroje používané pro měření koncentrací vysoko nad limitními hodnotami
[9] ČSN EN 45544-4: Část 4: Pokyny pro volbu, instalaci, použití a údržbu
[10] ČSN EN 60079-29-1: Výbušné atmosféry – Část 29-1: Detektory plynů – Funkční požadavky na

detektory hořlavých plynů
[11] ČSN EN 60079-29-2: Výbušné atmosféry – Část 29-2: Detektory plynů – Výběr, instalace, použití a údržba detektorů hořlavých plynů a kyslíku

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Petr Olivka, Ph.D.**

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016

VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2016 ..... Hehlík Lukáš

Na tomto místě bych chtěl poděkovat firmě ZAM-Servis s.r.o. za návrh tématu a prostředky k uskutečnění práce, Ing. Václavu Žvakovi za konzultace ohledně výpočtů koeficientů a komunikace MODBUS, Ing. Petru Olivkovi, Ph.D. za vedení práce a připomínky k práci a ostatním, kteří mi pomohli nebo mě podpořili v práci.

Abstrakt

Diplomová práce se zabývá návrhem a implementací programu určeného ke komunikaci mezi sítí detektorů plynů a řídícím počítačem pomocí standardu RS485 a protokolu MODBUS. Práce se postupně věnuje použitým technologiím, základům detektorů plynů a implementaci programu. Výsledkem práce je program pro kalibraci detektorů plynů, praktické ověření funkčnosti a stability programu reálnou kalibrací a možnosti běhu programu na systému Linux.

Klíčová slova: C#, RS485, detektory plynů, kalibrace, Mono, linearizace

Abstract

This master's thesis deals with design and implementation of application used for communication between network of gas detectors and personal computer via RS485 standard and MODBUS protocol. Thesis describes used technologies, basics of gas detectors and implementation of application. The results of thesis are application for gas detectors calibration, practical verification of functionality and stability of application by detector calibration and capability to run application under Linux based operating systems.

Key Words: C#, RS485, gas detectors, calibration, Mono, linearization

Obsah

Seznam použitých zkratek a symbolů

AD	–	Analog / Digital
ADU	–	Application Data Unit
ASCII	–	American Standard Code for Information Interchange
BSD	–	Berkeley Software Distribution
CR	–	Carriage Return
CRC	–	Cyclic Redundancy Check
ČSN EN	–	Česká Státní Norma Evropská Norma
DA	–	Digital / Analog
ECMA	–	European Computer Manufacturers Association
FRAM	–	Ferroelectric Random Access Memory
IP	–	Internet Protocol
ISO	–	International Organization for Standardization
LF	–	Line Feed
LEL	–	Lower Explosive Limit
LRC	–	Longitudinal Redundancy Check
Mbit	–	megabit
OS	–	Operační Systém
PC	–	Personal Computer
ppm	–	Parts per million (z angličtiny, česky „dílů či částic na jeden milion“), zkráceně též ppm, je výraz pro jednu miliontinu (celku). Používaný často pro udávání objemové koncentrace toxických plynů.
PDU	–	Protocol Data Unit
PLC	–	Programmable Logic Controller
SP3	–	Service Pack 3
TCP	–	Transmission Control Protocol
XML	–	Extensible Markup Language

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Detektory plynů musí splňovat řadu nařízení a norem, a tak jako ostatní zařízení postupem času stárnou a mění se jejich citlivost detekovat danou látku. Z tohoto důvodu musí být detektory kalibrovány v určitých intervalech, aby se zaručila jejich správná funkce rozeznávání koncentrace plynů. Obsluhující pracovník musí během kalibračního procesu předávat detektoru pokyny a kalibrační data. Pro tento účel se využívá aplikace, která zprostředkovává výpočty a komunikační rozhraní mezi pracovníkem provádějícím kalibraci a samotným detektorem.

Vzhledem k mé dlouhodobé spolupráci s firmou ZAM-SERVIS s.r.o., jsem byl firmou osloven s návrhem tématu pro diplomovou práci. Přesněji k realizaci programového vybavení určeného pro servis a kalibraci senzorů plynů, které firma vyrábí a servisuje. Hlavním cíle práce je vytvořit plnohodnotnou náhradu stávajícího programu, který v dnešní době již nevyhovuje po více stránkách.

2 Stávající řešení a kalibrace přístrojů

Kapitola shrnuje důvody vzniku práce a požadavky na nové programové vybavení. Podkapitoly seznamují čtenáře s procesem kalibrace a důležitými pojmy vztahujícími se k tomuto procesu.

2.1 Stávající program a nové požadavky

Stávající servisní program je zastaralý, jak po stránce technické, tak po stránce aktuálnosti a funkcionalit. Program byl vyvinut ve vývojovém prostředí Borland C++ Builder 6 z roku 2002, které již není výrobcem podporováno a je obtížné jej používat v nových verzích operačního systému. Samotný program je napsán nevhodným způsobem a jeho modifikace je obtížná.

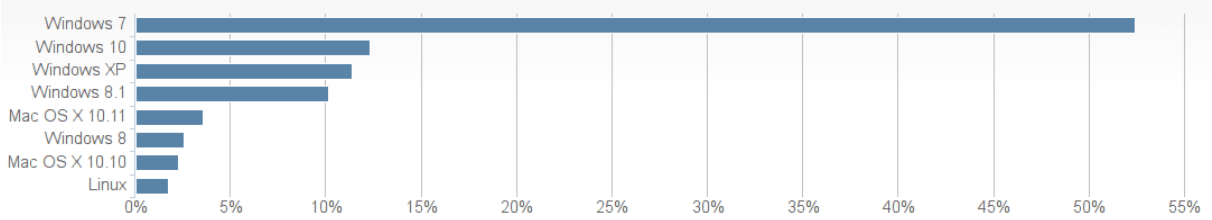
Program umožňuje v jeden okamžik kalibrovat pouze jeden detektor, to znamená, že pro detektor se musí provést celý kalibrační proces a poté pokračovat dalším detektorem. Tento postup je tak časově náročný a neekonomický, jelikož při každé změně plynu se musí v případě použití míchacího zařízení plynů pročistit (vypláchnout) potrubí od zbytků předchozího plynu a zaplynovat novým aktuálně používaným plynem. Takto se zbytečně plýtvá časem pro neustále vyplachování potrubí a také se zvyšují náklady na kalibraci, protože v potrubí vždy zůstane plyn, který se následně musí vypláchnout. Pokud by program podporoval možnost kalibrace více zařízení najednou, došlo by ke snížení spotřeby plynu a potřebného času na kalibraci více detektorů. Tímto by se nemusel dokončit celý kalibrační proces pro jeden detektor.

Z těchto důvodů vznikl návrh na vytvoření nové verze programu v jazyce C# rozšířené o další funkcionality, aby se zejména eliminovaly problémy běhu programu na nových verzích operačního systému Windows a doplnění o nové funkcionality umožnující provádět prvotní kalibraci detektorů s infračervenými senzory. Programovací jazyk C# je vybrán firmou, jelikož tento jazyk běžně používají a mají k němu vývojové prostředí. Požadavky na nové programové vybavení jsou:

- programovací jazyk C#,
- kompatibilita s OS Windows od verze XP,
- jazykové varianty programu s možností vytváření překladů uživatelem,
- výpočet kalibračních konstant pro linearizaci výstupů pro senzory, které nemají lineární výstup,
- výpočet kompenzačních konstant,
- podpora kalibrace více detektorů najednou,
- výpočet hodnot T90, T50, T10, T63,

Ačkoliv první požadavek, kompatibilita s OS Windows od verze XP, může vznášet otázky proč podporovat systém Windows XP, který již není podporován (podpora ukončena 8. 4. 2014,

podpora Embedded 12. 1. 2016), tak odpověď je jednoduchá. Operační systém Windows XP je stále v praxi hojně zastoupen (11 %) a využíván, ať už se jedná o použití ve firmách nebo v embedded systémech (Embedded systémů se práce netýká). Graf 1 znázorňuje zastoupení systému Windows XP na trhu (graf převzat z článku [36]). Aby bylo docíleno podpory i systému Windows XP, musel jsem použít .NET Framework maximálně verze 4.0, který je nejvyšší dostupnou verzí pro Windows XP (Service Pack 3) [36, 42].



Obrázek 1: Podíl systému Windows XP na trhu.

2.2 Kalibrace přístrojů

Nejprve podle TNI 01 0115 (Mezinárodní metrologický slovník) uvedu několik základních pojmů: [38, 39]

- *Kalibrace* – činnost, která za specifikovaných podmínek v prvním kroku stanoví vztah mezi hodnotami veličiny s nejistotami měření poskytnutými etalony a odpovídajícími indikacemi s přidruženými nejistotami měření a ve druhém kroku použije tyto informace ke stanovení vztahu pro získání výsledku měření z indikace.
- *Justování* – soubor činností provedených na měřicím systému tak, aby poskytoval předepsané indikace odpovídající daným hodnotám veličiny, která má být měřena.
- *Ověřování* – poskytnutí objektivního důkazu, že daná položka splňuje specifikované požadavky.
- *Ověření kalibrace* – činnost, která následuje po kalibraci. Nejsou-li splněny specifikované požadavky měřidla, je to signál k opakování procesu kalibrace případně justování a opakovanému ověření kalibrace. Do kategorie pojmu ověření kalibrace lze zařadit vykonání zkoušky funkčních (metrologických) parametrů měřidla, které má mít po provedené kalibraci. Mezinárodně zavedené slovní spojení „ověření kalibrace“ (verification of calibration) může zdánlivě národním prostředí kolidovat se slovním spojením „ověření stanoveného měřidla“, ale každé má rozdílný význam.

V tomto textu je pojem kalibrace používán jako souhrnné označení pro soubor činností a postupů zahrnujících kalibraci, justování, ověřování, a to i opakovaně, vedoucích ke stavu, kdy měřidlo má všechny indikace v požadovaných mezích. Takto je tento pojem „kalibrace“

uplatňován také při činnostech, které jsou prováděny při výrobě snímačů koncentrace plynů ve firmě ZAM-SEVIS s.r.o.

Pojmem *prvotní kalibrace*, je kalibrace provedená při výrobě, zahrnující i stanovení a zadaní linearizačních konstant u snímačů s infračerveným senzorem.

2.2.1 Proces kalibrace

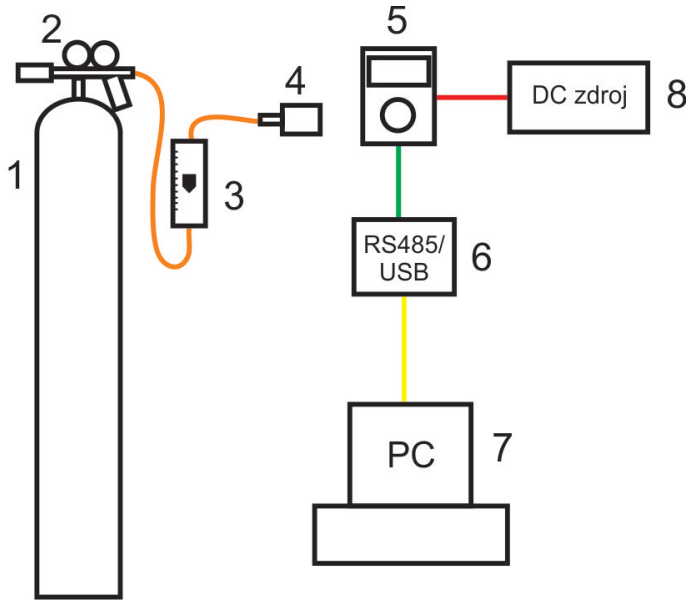
Kalibrace detektoru plynu se provádí vystavením senzoru danému plynu ve dvou nebo několika různých koncentracích. Nejprve se detektor kalibruje takzvaným *nulovým plynem*, jedná se plyn nebo směs, který má nulovou koncentraci detekovaného plynu. Nejčastěji se jako nulový plyn využívá syntetický vzduch, což je směs kyslíku a dusíku v poměru 20 % O_2 a 80 % N_2. Dále se používají *kalibrační plyny*, které jsou co nejpřesněji namíchány k požadované koncentraci a mají status etalonu. Více kalibračních plynů se využívá při provádění vícebodové kalibrace, kdy je nutné provést měření pro několik různých hodnot koncentrací plynu, aby mohl být výstup nelineárního senzoru zkalibrován a zlinearizován pro následující provozní kalibrace.

Kalibrační sestava je znázorněna ilustrací 2 a skládá se z:

- 1 - tlaková láhev s nulovým nebo kalibračním plynem,
- 2 - vícestupňový regulační ventil,
- 3 - průtokoměr,
- 4 - kalibrační nástavec,
- 5 - detektor plynu,
- 6 - převodník RS485/USB,
- 7 - PC,
- 8 - zdroj stejnosměrného napětí.

Oranžovou barvou je vyznačena hadička, ve které proudí plyn spojující regulační ventil, průtokoměr a nástavec. Zeleně je vyznačena sériová linka RS485. Žlutá barva reprezentuje propojení PC a převodníku RS485/USB USB kabelem. Červená linka znamená napájecí vedení detektoru plynu. Regulační ventil a průtokoměr může být také součástí míchacího zařízení plynů využívaného v servisních střediscích.

Tuto kalibrační sestavu používá pracovník v servisním středisku pro prvotní nebo provozní kalibraci detektorů plynů. Proces kalibrace probíhá následovně: pracovník si připraví kalibrační pracoviště s tlakovými láhvemi obsahující potřebné plyny pro kalibraci, připojí k detektoru plynu napájení a sériovou linku. Servisním programem v osobním počítači naváže komunikaci s detektorem, kterému programem odešle příkaz indikující začátek kalibračního procesu. Následně pracovník vystaví detektor nulovému plynu, vyčká na ustálení a odešle programem detektoru



Obrázek 2: Kalibrační sestava.

informaci o přítomnosti nulového plynu. Poté pokračuje s kalibračním plynem, opět vystaví detektor plynu a odešle programem informaci o aplikaci kalibračního plynu s příslušnou koncentrací. Zařízení si přijatá data zpracuje, uloží do paměti. Při výpočtu koncentrace zařízení dosadí data z paměti a hodnoty z AD převodníku do výpočetní rovnice a vypočte hodnotu koncentrace. Provádí-li se vícebodová kalibrace, tak se proces opakuje kolikrát je požadováno kalibračním předpisem. Celý proces se opakuje pro každý kalibrovaný detektor.

2.2.2 Časy odezvy při kalibraci

Během kalibrace se aplikují plyny různých koncentrací. Většinou je doprava plynu ke snímacím elementům zajištěna difuzí, tedy samovolným pronikáním plynu. Proto, pokud přivedeme zkušební plyn ke snímači, stoupá indikovaná koncentrace po exponenciální křivce tak, jak se postupně zvyšuje obsah plynu v blízkosti měřících elementů. Vzhledem k tomuto průběhu, nemůže koncentrace na senzoru dosáhnout skutečné koncentrace plynu. Proto jsou stanoveny časy, které se používají pro vyjadřování vlastností senzoru. T90, T63, T50, T10, tyto časy jsou specifikovány jako odezva na jednotkový skok, a dosažení hodnot velikosti tohoto skoku v procentech. Graf na obrázku 3 zobrazuje vztah času a koncentrace při jednotkové změně [1, 2].

T90, vyjadřuje rychlost odezvy na jednotkový skok. Toto je také důležitý parametr pro kalibraci. Pro kalibraci se považuje, že za dobu 3krát T90 je hodnota natolik ustálená, že další její vzrůst je z metrologického hlediska nevýznamný. Má-li snímač udaný čas T90 30 sekund, tak při přivedení kalibračního plynu, lze po 90 sekundách indikovaný údaj považovat za ustálený [1, 2].

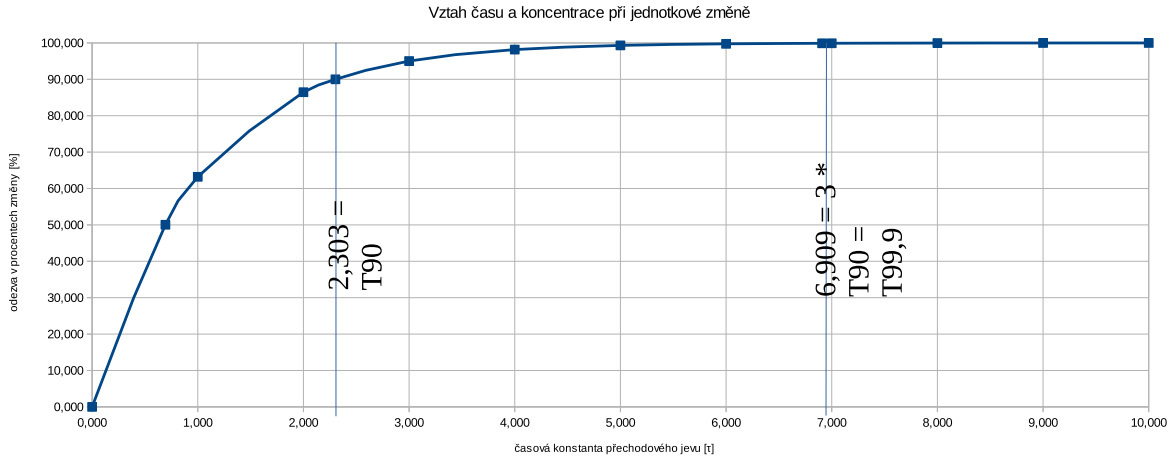
T63, toto je čas, který lze považovat za jednotku na exponenciále, zpravidla T90 je 2,3krát delší než T63. Jde o pomocný čas, který má souvztažnost s T90, a lze pomocí něj odhalit případná

nevhodná dopravní zpoždění. V běžné technické praxi se tento údaj nepoužívá. Případně lze tento údaj nalézt u teplotních snímačů, kde se uvádí místo T90 [1, 2].

T50, čas používaný při laboratorním posouzení snímače, při certifikaci. V běžné technické praxi se tento údaj nepoužívá.

T10, čas inverzní k T90, udává čas za který koncentrace poklesne na 10 procent jednotkového skoku. Většinou se udává pouze u kombinace plynů a snímačů, které se špatně odvětrávají, například těžké plyny [1, 2].

Orientačně lze říci, že čas pro dosažení hodnoty T63 je 1,0, pak čas pro T90 je 2,3 a času 3xT90 odpovídá hodnota T99,9.



Obrázek 3: Vztah času a koncentrace při jednotkové změně.

2.2.3 Parametry ovlivňující kalibraci

Indikovanou hodnotu koncentrace mohou při kalibraci ovlivnit: okolní teplota, tlak vzduchu a rychlost proudění plynu, případné další faktory nejsou tak významné. Snímače jsou kompenzovány na teplotu a pokud je použit předepsaný kalibrační nádstavec, tak je minimalizovaný i vliv tlaku. Pokud je použit předepsaný kalibrační nádstavec a výrobcem předepsané množství plynu je eliminován i vliv proudění plynu na senzor. Zvyšování množství kalibračního plynu sice mírně zkracuje čas T90, ale výrazně zvyšuje spotřebu kalibračního plynu, který s ohledem na to, že jde o referenční materiál je velmi drahý. Doporučené množství kalibračního plynu se pohybuje mezi 0,5 až 1 litrem za minutu.

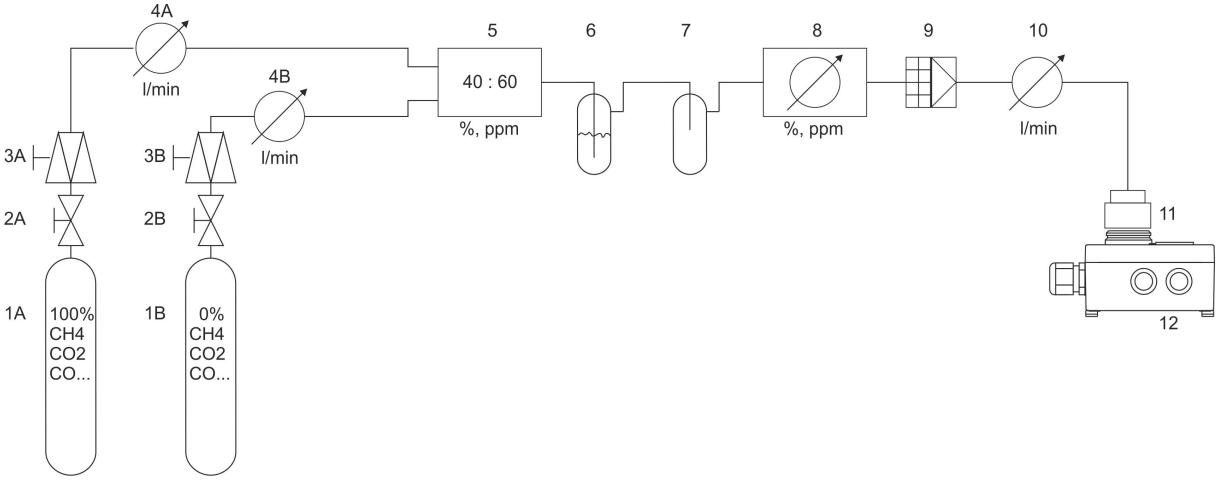
2.2.4 Ekonomika kalibrace

Plyny používané pro kalibraci musí mít, v rámci metrologie, status referenčního materiálu. Jde o materiál, který je dostatečně homogenní a stabilní, s referencí ke specifikovaným vlastnostem,

které byly stanoveny tak, že se hodí pro jejich zamýšlené použití při měření nebo při zkoumání jmenovitých vlastností.

Příprava takových kvalitních materiálů a jejich ověřování je poměrně náročná na odbornost a vybavení. Tomu pak odpovídá cena, například 10 litrová láhev obsahující plyn s 2% koncentraci metanu stojí cca 5 000,- Kč. Z takovéto láhve získáme pouze cca 1,5 m^3, toto odpovídá spotřebě zhruba za 3 směny. Pro kalibraci snímačů, které mají senzory s lineární odezvou, stačí jeden kalibrační plyn s hodnotu cca 80 % měřicího rozsahu a nulový plyn. Pro kalibraci snímačů, které používají senzory pracujících na infračerveném principu je nutno při prvotní kalibraci použít nejméně 3 plyny různých koncentrací vhodně rozložených v měřícím rozsahu přístroje.

Pro snížení ceny za kalibrační plyny se používají vysoko koncentrované plyny většinou se 100 % koncentrací měřeného plynu. Cena takového plynu se příliš neliší od ceny kalibračního plynu potřebné koncentrace. Tyto plyny se pak ředí pomocí objemových ředičeк plynů na potřebnou koncentraci, přesnost těchto ředičeк bývá o řád lepší než nejistota referenčního materiálu, pro jistotu se však do měřícího řetězce vřazuje laboratorní spektrální analyzátor. Obrázek 4 znázorňuje kalibrační sestavu s objemovou ředičkou plynů skládající se z: 1A - tlaková láhev s kalibračním plynem 100% koncentrace; 2B - tlaková láhev s nulovým plynem; 2 - uzavírací ventil láhve; 3 - redukční ventil; 4 - indikátor průtoku, průtokoměr; 5 - ředička plynů; 6 - zvlhčování plynu, dle potřeby; 7 - odlučovač, dle potřeby; 8 - laboratorní analyzátor, měření koncentrace plynu; 9 - protiexplozívní pojistka s ventilem; 10 - průtokoměr, 0,5 l/min; 11 - kalibrační nádstavec; 12 - kalibrovaný snímač.



Obrázek 4: Objemová ředička plynů.

Pokud střídáme v kalibračním systému různé koncentrace plynu je nutno vždy počkat, až se celý systém propláchne novou koncentrací plynu, a teprve poté provádět vlastní kalibraci. Proto je výhodné kalibrovat několik snímačů najednou a tím omezit mrtvé časy čekání na propláchnutí systému novou koncentrací plynu.

Jedním z úkolů programu je právě umožnit kalibraci více snímačů najednou. Snímače umožňují sice řídit kalibraci pomocí tlačítek na snímači, ale pokud se pro řízení využije program

komunikující se snímačem pomocí sériového rozhraní je ovládání a zadávaní hodnot pohodlnější, rychlejší a spolehlivější. U snímačů vybavených infračervenými senzory nejdou výchozí kalibrační koeficienty dokonce ani nijak jinak zadat a vypočítat, než pomocí počítače.

2.2.5 Interval kalibrace

Stanovení intervalu kalibrace je poměrně složitý problém do kterého vstupuje mnoho činitelů. Základní orientaci o tomto intervalu lze nalézt například v normě ČSN EN 45544-4. Tento interval je dán zejména konstrukcí, okolním prostředím, zkušeností s provozem a legislativními požadavky. Měření plynů je, jako každé jiné měření, podřízeno metrologické legislativě včetně stanovení intervalu a způsobu kalibrace, a mělo by být začleněno do metrologického řádu organizace [1, 2].

Výrobce: Na základě konstrukce, použitých senzorů, materiálů, stupně SIL a dalších parametrů, stanoví výrobce základní kalibrační interval, během kterého se předpokládá, že nedojde ke zhoršení kvalitativních a bezpečnostních parametrů snímače při předpokládaném způsobu používaní.

Okolní prostředí: Podle okolního prostředí může dojít ke zkrácení intervalu, například při zvýšené prašnosti, extrémní vlhkosti, přítomnosti materiálu způsobujících otravu senzoru.

Uživatel: Podle vlastní zkušenosti s provozem konkrétních snímačů v konkrétním prostředí a klimatu, může uživatel zkrátit nebo prodloužit interval kalibrace, a to i podle místa nasazení konkrétního snímače. Nikdy však nemohou být stanoveny intervaly delší než jsou intervaly požadované legislativou.

Legislativa: Podle konkrétního použití a aplikace se mohou na snímač vztahovat různé legislativní požadavky, a to i co se týče intervalu kalibrace. Například konkrétní intervaly kalibrací jsou stanoveny vyhláškami a normami pro plynové kotelny, podle typu kotelny jsou například stanoveny intervaly kalibrace na 6 měsíců s měsíční kontrolu funkčnosti. Jiné intervaly jsou stanoveny pro aplikaci která spadá do působnosti statní báňské správy, tam je to řešeno báňskými předpisy, zde mohou být stanoveny i intervaly 1 až 2 týdny podle expozice pracoviště prachem nebo metanem.

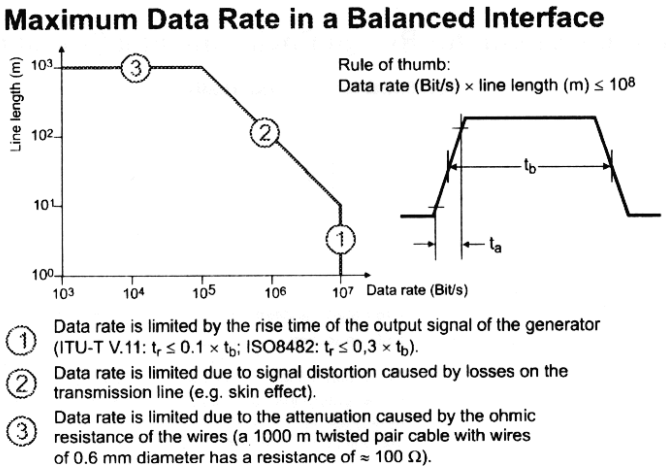
Obecně lze říci že, pokud na konkrétní aplikaci je uplatněn nějaký předpis, dokument nebo návod který stanoví interval kalibrace, musí se dodržet ten interval, který je nejkratší.

U snímačů SC- firmy ZAM-SERVIS s.r.o. je stanoven v uživatelské příručce kalibrační interval jeden měsíc, tento interval zohledňuje zejména nutnost udržení kontroly nad snímačem zabezpečujícím bezpečnost a zohledňující předpokládané použití v náročném prostředí. Pokud se snímače SC- použijí v „kancelářském prostředí“, jsou podle typu senzoru schopny udržet své parametry i déle než rok.

3 Komunikace s přístroji

3.1 RS485

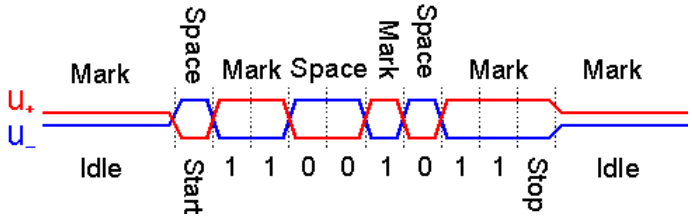
RS485 je průmyslová komunikační sběrnice využívaná především v průmyslových aplikacích nebo v prostředích s požadavky na vysokou odolnost proti rušení. Sběrnice je obdobou rozhraní RS232, od kterého se liší především jinými napěťovými úrovněmi, nepřítomností pomocných řídících signálů, možností multipoint komunikace a symetrickými signály, které zvyšují odolnost proti rušení a umožňují dosáhnout velkých přenosových vzdáleností. Sběrnici RS485 lze vytvořit z rozhraní RS232 pomocí převodníků úrovně. RS485 využívá pro komunikaci kroucený pár, kde jeden vodič je označován písmenem A (invertující pin, který je negativní, když je vedení nečinné) a druhý vodič písmenem B (neinvertující pin, který je pozitivní, když je vedení nečinné), vodiče někteří výrobci označují opačně a někteří také jako RxTx- a RxTx+. Maximální délka sběrnice je 1200 m a lze ji prodloužit za použití opakovačů. Přenosová rychlost je nepřímo úměrná délce vedení a u krátkých vedení lze dosáhnout rychlosti do 10 Mbit/s. Závislost přenosové rychlosti na délce vedení popisuje obrázek 5 (obrázek převzat z článku [3]). Standard definuje použití maximálně 32 zařízení na jedné větvi. Zařízení musí být zapojována za sebou, nikoliv do hvězdy. Komunikace na lince funguje v poloduplexním režimu, to znamená, že v danou chvíli může vysílat pouze jedno zařízení (systém dotaz-odpověď) [33, 3].



Obrázek 5: Závislost přenosové rychlosti na délce vedení RS485.

Stejně jako RS232 využívá RS485 dvě napěťové úrovně pro komunikaci s tím rozdílem, že logické úrovně jsou reprezentovány rozdílem napětím mezi oběma vodiči a nejsou vztaženy k referenční zemi. Tímto systémem rozeznávání logických stavů je potlačena možnost rušení, jelikož indukovaný rušivý signál se většinou přičítá k oběma vodičům stejně. Logické stavy jsou detekovány, pokud rozdíly mezi vodiči A a B jsou větší jak 200 mV nebo menší jak -200 mV. Logické stavy jsou označovány stejně jako u RS232, logická 1 označována jako „mark“ a logická 0 jako „space“. Pro přenos informací se využívá 7 nebo 8bitový rámec se start bitem, jedním

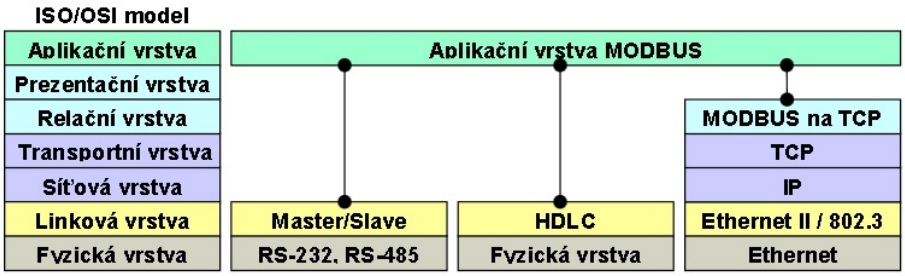
nebo zdvojeným stop bitem a volitelným paritním bitem. Podobně jako u RS232 je start bit realizován logickou 0 tedy „space“ a stop bit logickou 1 tedy „mark“. Na obrázku 6 (obrázek převzat z článku [33]) je vyobrazen přenos rámce RS485 s vyznačeným průběhem napěťových úrovní [33, 3].



Obrázek 6: Rámec RS485.

3.2 MODBUS

MODBUS je otevřený komunikační protokol pracující na úrovni aplikační vrstvy ISO/OSI modelu pro vzájemnou komunikaci typu klient-server různých zařízení přes různé typy sítí a sběrnic, například sériové linky, rádiové a optické sítě a síť Ethernet s protokolem TCP/IP. Možnosti implementace protokolu je ilustrována na obrázku 7 (obrázek převzat z článku [22]). Protokol byl vytvořen firmou Modicon v roce 1979. Na komunikační síti se vyskytuje jedno zařízení typu klient (označované též master, v případě TCP/IP může být těchto zařízení více) a 1 až 247 (v případě RS485) zařízení typu server (slave). Komunikace probíhá dotazováním ze zařízení typu klient na zařízení typu server, které odpovídá na dotazy adresované pouze jemu. Roli klienta zastupuje řídící zařízení, například PLC nebo PC. Roli serveru mají ovládané nebo sledované zařízení, například detektory, měřící přístroje, PLC a další [32, 5, 22].

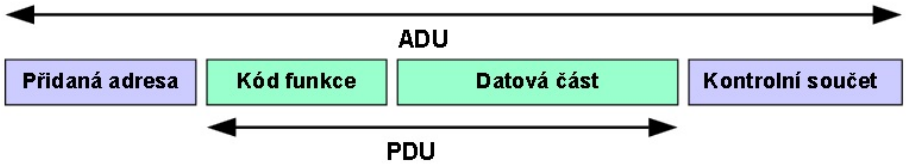


Obrázek 7: Implementace protokolu MODBUS.

Struktura PDU (Protocol data unit) je definována protokolem nezávisle na druhu použité komunikační vrstvy, což dovoluje používat protokol MODBUS přes různé komunikační sítě a sběrnice. PDU se rozšiřuje podle použitého typu komunikačního prostředí o další součásti a vytváří ADU (Application data unit). Struktura PDU a ADU je vyobrazena na ilustraci 8 (obrázek převzat z článku [22]). Protokol definuje tři typy PDU zpráv: [32, 5, 22]

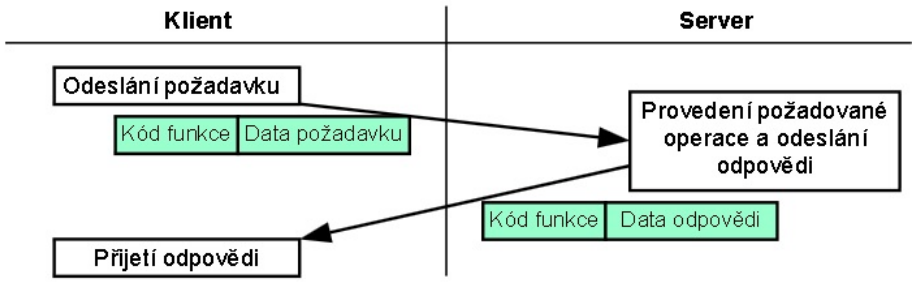
- požadavek (Request PDU),

- odpověď (Response PDU),
- záporná odpověď (Exception Response PDU).

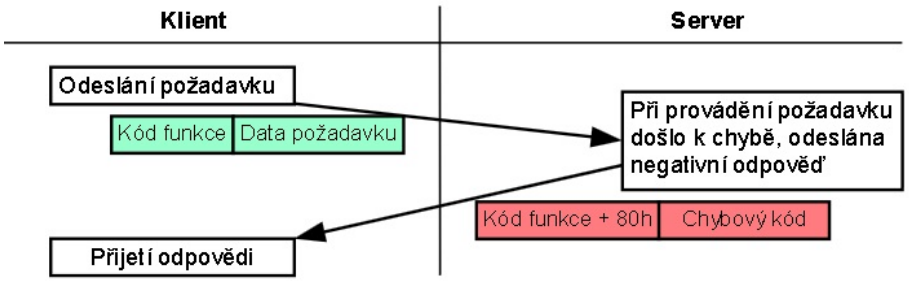


Obrázek 8: Struktura MODBUS PDU a ADU.

Pokud během zpracovávání operace serverem nedošlo k chybě, odpoví server klientovi zprávou obsahující v kódu funkce kód vykonané funkce k indikaci úspěšného provedení funkce. Vyžadovali klient data po serveru, tak jsou mu požadovaná data vrácena v poli data. Kdyby se během zpracování požadované operace vyskytla chyba, je v poli kódu funkce vrácen kód vykonávané funkce, u kterého je nastaven nejvyšší bit na logickou jedničku a v datové části je vrácen chybový kód (tabulka 1). Pro lepší orientaci jsou zmíněné případy ilustrovány obrázky 9, 10 (obrázky převzaty z článku [22]) [32, 5, 22].



Obrázek 9: MODBUS komunikace bez chyby.



Obrázek 10: MODBUS komunikace s chybou.

Obsah zprávy jsou samotná data předávané mezi zařízeními, data mohou obsahovat adresu a počet vstupů, které má server přečíst, hodnotu registrů pro zápis, nebo může být datová část prázdná. Kód funkce určuje jakou operaci má server vykonat, přehled základních kódu je v tabulce 2. Rozsah kódů je 1 až 255, kde 128 až 255 jsou vyhrazeny pro oznámení chyby (záporné odpovědi). Kontrolní součet je CRC pro RTU Mode a LRC pro ASCII Mode [32, 5, 22].

Tabulka 1: Kódy chyb protokolu MODBUS.

Kód	**Název chyby**	**Popis**
01	Chybná funkce	Požadovaná funkce není zařízením podporována.
02	Chybná adresa dat	Zadaná adresa je mimo podporovaný rozsah.
03	Chybná hodnota dat	Předávaná data jsou neplatná.
04	Selhání zařízení	Při provádění požadavku došlo k neodstranitelné chybě (požadavek nelze nyní provést).

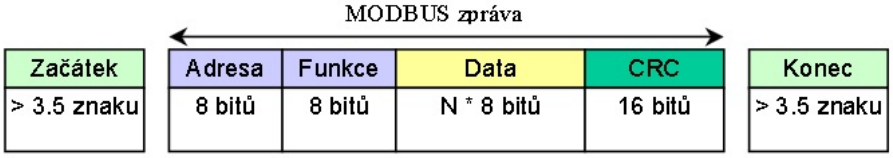
Tabulka 2: Kódy základních funkcí protokolu MODBUS.

Kód	**Název funkce**	**Popis**
01	Čti cívky	Čtení jednoho nebo více bitů.
02	Čti diskrétní vstupy	Čtení jednoho nebo více bitů.
03	Čti uchovávací registry	Čtení jednoho nebo více 16bitových registrů.
04	Čti vstupní registry	Čtení jednoho nebo více 16bitových registrů.
05	Zapiš jednu cívku	Zápis jednoho bitu.
06	Zapiš jeden registr	Zápis jednoho 16bitového registru.
15	Zapiš více cívek	Zápis více bitů.
16	Zapiš více registrů	Zápis více 16bitových registrů.

Pro komunikaci po sériové lince (RS232, RS485) je protokol striktně v režimu klient-server, v jeden okamžik může být na sběrnici pouze jeden klient a až 247 serverů. Komunikaci zahajuje vždy pouze klient a server nikdy nesmí vysílat bez pověření klienta. Klient posílá zprávy (požadavky) serverům v režimech unicast a broadcast, režimy se chovají podobně jako u TCP/IP. V unicast režimu klient adresuje zprávu jednomu konkretnímu serveru a ten odpoví klientovi. V broadcast režimu klient odešle zprávu všem serverům, ale žádný server mu neodpoví, jako broadcast adresa se využívá 0. Adresování serverů je v rozsahu 1 až 247, tato adresa musí být v síti jedinečná, klient adresu nemá. Protokol definuje dva vysílací režimy pro komunikaci po sériové lince, jedná se o RTU Mode a ASCII Mode. Režim určuje v jakém formátu jsou data vysílána a jak jsou dekódována. Na jedné síti musí všechny zařízení používat stejný režim vysílání. Zařízení musí implementovat RTU Mode oproti ASCII Mode, který je nepovinný [32, 5, 22].

V RTU Mode se byte zprávy vysílá jako dva 4bitové hexadecimální znaky. Vysílaní musí být souvislé a mezery mezi znaky nesmí být delší než 1,5 znaku. Pomlka na sběrnici ve větší délce než 3,5 znaku identifikuje začátek a konec zprávy. 16bitové pole CRC ve zprávě slouží k detekci chyb. Zařízení musí podporovat sudou paritu, v případě, že parita není použita, tak je nahrazena druhým stop bitem. RTU Mode rámec znázorňuje obrázek 11 (obrázek převzat z článku [22]) [32, 5, 22].

V ASCII Mode je každý byte zprávy vysílán jako dvojice ASCII znaků. V porovnání s RTU Mode je pomalejší, ale dovoluje vysílat znaky s prodlevami až 1 sekundu. Začátek zprávy je identifikován znakem „:“ a konec řídícími znaky CR, LF. Zařízení musí podporovat sudou paritu, v případě, že parita není použita, tak je nahrazena druhým stop bitem. Oproti RTU Mode je tato



Obrázek 11: MODBUS RTU Mode rámec.

verze méně využívaná i když je pro člověka čitelnější. ASCII Mode rámec znázorňuje obrázek 12 (obrázek převzat z článku [22]) [32, 5, 22].

Začátek	Adresa	Funkce	Data	LRC	Konec
znak „:“	2 znaky	2 znaky	0 až 2*252 znaků	2 znaky	2 znaky CR, LF

Obrázek 12: MODBUS ASCII Mode rámec.

3.3 Komunikace

Detektory plynů implementují komunikaci protokolem MODBUS, ze kterého využívají pouze funkce 3, 6 a 16 (viz. tabulka 2), ostatní funkce nejsou zařízeními podporovány. Výrobcem je stanovený limit dotazování na zařízení jednou za sekundu, častější dotazování může ovlivnit funkčnost zařízení. Zápisem nevhodných hodnot, zápisem a čtením mimo implementované příkazy a adresy může dojít k nefunkčnosti zařízení. Během výroby jsou do zařízení při programování nahrány všechny potřebné hodnoty konfigurace, při samotné kalibraci nebo servisu se ze zařízení většina hodnot pouze vyčítá a zapisuje se pouze pár parametrů, kalibračních příkazů a kalibračních konstant. Popis komunikace a její vzor bude dále popsán následujícími tabulkami [31].

3.3.1 Jednoduché příkazy

1. Čtení uchovávacích registrů (konfigurace zařízení), čte 1 až 125 16bitových registrů z paměti FRAM. Požadavek na server je popsán tabulkou 3, odpověď serveru tabulkou 4 a chybová odpověď tabulkou 5 [31].

Tabulka 3: Požadavek čtení konfigurace na server.

Kód funkce	**Počáteční adresa (2B)**	**Počet registrů (2B)**
0x03	0x0000 až 0x1FFF	N = 1 až 125 (0x7D)

2. Čtení uchovávacích registrů (měřené veličiny a stavy zařízení), čte 8, 10, nebo 12 16bitových registrů. Požadavek na server je popsán tabulkou 6, odpověď serveru tabulkami 7, 8 a chybová odpověď tabulkou 9 [31].

Tabulka 4: Odpověď čtení konfigurace ze serveru.

Kód funkce	**Počet bytů (1B)**	**Hodnoty registrů (2xN)**
0x03	2xN	

Tabulka 5: Chybová odpověď při čtení konfigurace.

Kód funkce	**Chybový kód (1B)**
0x83	01, 02, 03, 04

Tabulka 6: Požadavek čtení veličin a stavů na server.

Kód funkce	**Počáteční adresa (2B)**	**Počet registrů (2B)**
0x03	0x8000	8, 10, 12

Tabulka 7: Odpověď čtení veličin a stavů ze serveru.

Kód funkce	**Počet bytů (1B)**	**ID zařízení (2B)**	**ID zařízení (2B)**	**Koncentrace (2B) Hodnota**	**0 (H) + Tepl(L) (2B)**	**3V napětí (2B)**
0x03	2xN	SignaturaH	SignaturaL	0 až 65535	0 až 255	0 až 655,35 V

Tabulka 8: Pokračování odpovědi čtení veličin a stavů ze serveru.

0 (H) + Nap (L) (2B)	**0 (H) + Stav (L) (2B)**	**ID senz. (H) + SW verze (L)**	**doplňková Veličina (2B)**	**0, NU**	**errors bits**	**0 (H) Čítač akt. hodnoty, (L) Čítač Rx paketů**
0 až 255 V	0 až 255	0 až 65535	+/-32000	0	0 až 65535	0 až 255, 0 až 255

Tabulka 9: Chybová odpověď při čtení čtení veličin a stavů.

Kód funkce	**Chybový kód (1B)**
0x83	01, 02, 03, 04

3. Zápis jednoho registru, spuštění měření po překročení rozsahu. Požadavek na server je popsán tabulkou 10, odpověď serveru tabulkou 10 a chybová odpověď tabulkou 11 [31].

Tabulka 10: Požadavek zápisu spuštění měření po překročení rozsahu.

Kód funkce	**Adresa registru (2B)**	**Hodnota registru (2B)**
0x06	0xFFF7	0xABCD

Tabulka 11: Chybová odpověď při zápisu.

Kód funkce	**Chybový kód (1B)**
0x86	01, 02, 03, 04

4. Zápis jednoho registru, maximální koncentrace tranzistorového výstupu. Požadavek na server je popsán tabulkou 12, odpověď serveru tabulkou 13 a chybová odpověď tabulkou 11 [31].

Tabulka 12: Požadavek zápisu maximální koncentrace tranzistorového výstupu.

Kód funkce	**Adresa registru (2B)**	**Hodnota registru (2B)**
0x06	0xFFF8	0 až 1500 (0 až 40000)

Tabulka 13: Odpověď zápisu maximální koncentrace tranzistorového výstupu.

Kód funkce	**Adresa registru (2B)**	**Hodnota registru (2B)**
0x06	0xFFF8	0 až 499 (0 až 40000)

5. Zápis jednoho registru, síťová adresa. Požadavek na server je popsán tabulkou 14, odpověď serveru tabulkou 14 a chybová odpověď tabulkou 11 [31].

Tabulka 14: Požadavek zápisu síťové adresy.

Kód funkce	**Adresa registru (2B)**	**Hodnota registru (2B)**
0x06	0xFFF9	0 až 247

Chybová odpověď je zde zmíněna pouze jednou a je na ni vícekrát odkázáno, jelikož jak víme z popisu protokolu MODBUS, tak chybová odpověď vždy obsahuje kód funkce, která se měla provést a chybový kód v části data. Odpovědi pro zápis spuštění měření po překročení rozsahu a síťové adresy rovněž nejsou zmíněny, ale odkázány na tabulku požadavku, jelikož jsou totožné s požadavkem.

3.3.2 Kalibrační procedura pro kompenzaci vlivu teploty na nulový signál

1. Nyní je nulový plyn při první teplotě (20 až 30 °C) [31]

   Tabulka 15: Požadavek zápisu do registru pro kompenzaci teploty při prvním plynu.

Kód funkce	**Adresa registru (2B)**	**Hodnota registru (2B)**
0x06	0xFFFA	0x0000

   Při odpovědi vrátí zařízení v části hodnota registru náhodné číslo, které musí klient zopakovat z důvodu kontroly.

   Tabulka 16: Odpověď zápisu do registru pro kompenzaci teploty při prvním plynu.

Kód funkce	**Adresa registru (2B)**	**Hodnota registru (2B)**
0x06	0xFFFA	0x0000 až 0xFFFF

2. Nyní je nulový plyn při druhé teplotě (alespoň o 20 °C vyšší než první) [31]

   V požadavku v poli hodnota registru se posílá náhodné číslo, které bylo přijato v předchozím kroku.

   Tabulka 17: Požadavek zápisu do registru pro kompenzaci teploty při druhém plynu.

Kód funkce	**Adresa registru (2B)**	**Hodnota registru (2B)**
0x06	0xFFFB	0x0000 až 0xFFFF

   Zařízení potvrdí kalibraci.

   Tabulka 18: Odpověď zápisu do registru pro kompenzaci teploty při druhém plynu.

Kód funkce	**Adresa registru (2B)**	**Hodnota registru (2B)**
0x06	0xFFFB	0x0000

3. Možnost zrušení kalibrační procedury [31]

   Tabulka 19: Požadavek zápisu zrušení kalibrace.

Kód funkce	**Adresa registru (2B)**	**Hodnota registru (2B)**
0x06	0xFFFC	0x0000 až 0xFFFF

   Zařízení potvrdí zrušení kalibraci. Případné chybové kódy jsou pouze 01 a 02.

3.3.3 Kalibrační procedura pro měření plynu

1. Chci kalibrovat, bude nulový plyn. [31]

Tabulka 20: Odpověď zápisu zrušení kalibrace.

Kód funkce	**Adresa registru (2B)**	**Hodnota registru (2B)**
0x06	0xFFFC	0x0000

Tabulka 21: Požadavek zápisu chci kalibrovat.

Kód funkce	**Adresa registru (2B)**	**Hodnota registru (2B)**
0x06	0xFFFF	0x0000

Při odpovědi vrátí zařízení v části hodnota registru náhodné číslo, které musí klient zopakovat z důvodu kontroly.

Tabulka 22: Odpověď zápisu chci kalibrovat.

Kód funkce	**Adresa registru (2B)**	**Hodnota registru (2B)**
0x06	0xFFFF	0x0000 až 0xFFFF

2. Nyní je nulový plyn, bude zkušební plyn [31]

   V požadavku v poli hodnota registru se posílá náhodné číslo, které bylo přijato v předchozím kroku.

Tabulka 23: Požadavek zápisu je nulový plyn.

Kód funkce	**Adresa registru (2B)**	**Hodnota registru (2B)**
0x06	0xFFFE	0x0000 až 0xFFFF

Tabulka 24: Odpověď zápisu je nulový plyn.

Kód funkce	**Adresa registru (2B)**	**Hodnota registru (2B)**
0x06	0xFFFE	0x0000

3. Nyní je zkušební plyn s určenou koncentrací [31]

   Hodnota registru obsahuje koncentraci plynu 0,00 % až 15,00 %, nebo 0 až 4000 ppm. Pokud je hodnota nula, pak se potvrzuje přítomnost nulového plynu.

Tabulka 25: Požadavek zápisu je zkušební plyn.

Kód funkce	**Adresa registru (2B)**	**Hodnota registru (2B)**
0x06	0xFFFD	0 až 1500 (0 až 40000)

Potvrzení kalibrace.

Během práce jsem se také seznámil s principy detektorů plynů a jejich měřících řetězců. V příloze A jsou zmíněny základy detektorů plynů a popisy různých typu senzorů.

Tabulka 26: Odpověď zápisu je zkušební plyn.

Kód funkce	**Adresa registru (2B)**	**Hodnota registru (2B)**
0x06	0xFFFD	0 až 500 (0 až 40000)

4 Návrh programu

4.1 Funkční požadavky

- **PROČ?** Stávající program je vyvinut v zastaralém vývojovém prostředí. Program má komunikační problémy na nových operačních systémech Windows a jeho rozšíření aktuálními požadavky je složité.
- **K ČEMU?** Potřebujeme servisní program pro komunikaci, servis, kalibraci, výpočty kalibračních a kompenzačních koeficientů detektorů plynů.
- **KDO?** Programové vybavení budou využívat odborní pracovníci v servisním středisku a ve výrobním procesu.
- **VSTUPY:** Údaje o použitých koncentracích plynů, teplotě okolí, konfigurační parametry přístrojů, nastavení časovačů a programu.
- **VÝSTUPY:** Zobrazování dat z registrů zařízení, zkalibrované zařízení, výsledky výpočtů kalibračních a kompenzačních koeficientů.
- **FUNKCE:** Detekce zařízení na sběrnici, provozní kalibrace detektorů, konfigurace dostupných parametrů programu. Výpis konfiguračních registrů, provádění provozní kalibrace detektorů, výpočet kalibračních a kompenzačních koeficientů při prvotní kalibraci, provádění vícebodové prvotní kalibrace pro více detektorů zároveň. Přehled proběhlých událostí programu, konfigurace přednastavení programu, jazykové překlady programu.

4.2 Nefunkční požadavky

- **PLATFORMA:** Program poběží na operačním systému Windows XP a vyšší.
- **IMPLEMENTACE:** Program bude napsán v jazyce C#.
- **SPOLEHLIVOST:** Program bude spolehlivě provádět proceduru kalibrace, zápis a čtení z přístrojů, výpočty kalibračních a kompenzačních koeficientů.
- **PŘEHLEDNOST:** Uživatelské rozhraní by mělo být přehledné a program by měl používat co nejméně formulářů.

4.3 Případy užití

Popsány jsou dva nejdůležitější případy užití, a to provozní kalibrace a vícebodová prvotní kalibrace.

Tabulka 27: Případy užití, kalibrace detektoru.

Název	Provozní kalibrace
ID	USE001
Primární aktér	Uživatel
Prekondice	Sériový port je otevřen
Scénář	1) Zobrazení karty pro kalibraci 2) Uživatel zvolí zařízení z tabulky 3) Uživatel požádá o začátek kalibrace 4) Program odešle požadavek na zařízení a zpracuje odpověď 5) Uživatel signalizuje přítomnost nulového plynu 6) Program odešle informaci do zařízení a zpracuje odpověď 7) Uživatel zadá koncentraci a signalizuje přítomnost kalibračního plynu 8) Program odešle kalibrační data do zařízení a zpracuje odpověď
Alternativní scénář	5) a 7) Uživatel zruší kalibrační proceduru
Výsledek akce	Detektor je zkalibrován na určitou koncentraci plynu

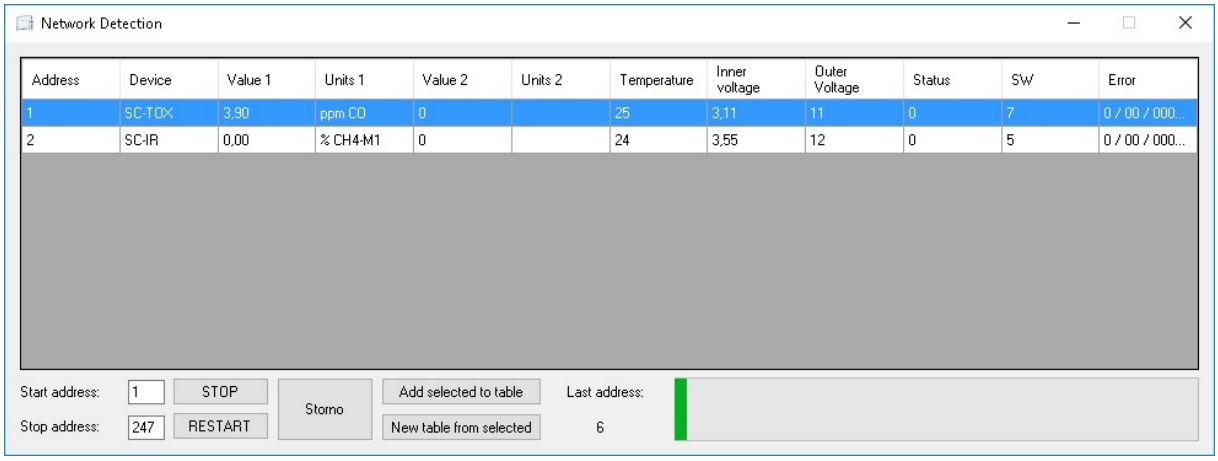
Tabulka 28: Případy užití, vícebodová počáteční kalibrace.

Název	Vícebodová počáteční kalibrace
ID	USE002
Primární aktér	Uživatel
Prekondice	Sériový port je otevřen
Scénář	1) Zobrazení karty pro prvotní kalibraci 2) Uživatel zvolí zařízení z tabulky 3) Uživatel zadá hodnoty koncentrace a teploty, zažádá o vyčtení hodnot ze zařízení 4) Program odešle požadavek na zařízení a zpracuje odpověď 5) Program zobrazí a uloží vyčtené hodnoty 6) Uživatel zvolí další zařízení z tabulky 7) Opakují se kroky 3 až 6 podle počtu zařízení a koncentrací plynu 8) Uživatel spustí výpočty koeficientů 9) Program provede výpočty a zobrazí výsledky 10) Uživatel zvolí koeficienty k zápisu do zařízení 11) Program odešle data do zařízení
Alternativní scénář	3) Uživatel zadá hodnoty koncentrace, teploty a intervalu periodického vyčítání. Spustí periodické vyčítání hodnot 4) Program vyčte data a zobrazí je, krok se opakuje dokud není zrušeno periodické vyčítání 5) Uživatel vyžádá nový údaj nebo provede výběr jiného zařízení 6) Program zpracuje požadavek dle bodu 5 7) Kroky 3 až 6 se opakuji podle počtu zařízení nebo koncentrací plynu
Výsledek akce	Výpočet kalibračních konstant

4.4 Popis uživatelského rozhraní

Vedlejším požadavkem bylo, aby program měl, co nejméně oken (formulářů) z důvodu zachování přehlednosti. Hlavní komponentou programu je formulář *FormCalibr*, ze kterého jsou případně volány další formuláře. V tomto hlavní formuláři se vyskytují veškeré funkce potřebné pro servis, provozní kalibraci senzorů, počáteční kalibraci a výpočty linearizačních koeficientů infračervených senzorů. V umístění spustitelného souboru se nachází konfigurační soubor programu, do kterého se ukládají přednastavené parametry programu. Zároveň se zde nachází složka *language* obsahující jazykové překlady programu. Program obsahuje celkem sedm formulářů, tyto grafické rozhraní jsou postupně popsány v této kapitole. Rozložení komponent grafického rozhraní bylo konzultováno a určeno firmou s požadavkem na podobnost ostatních programů firmy.

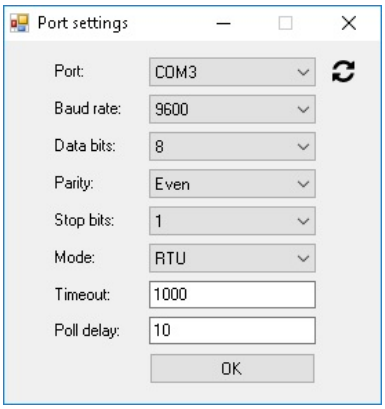
Pro komunikaci s detektory plynů potřebujeme znát jejich síťovou adresu, lze je tedy detekovat (skenovat) pomocí funkcionality formuláře FormNetworkDetection. Grafické rozhraní formuláře je zobrazeno na obrázku 13 a skládá se z prvků: tabulka pro zobrazení nalezených detektorů, ovládacích tlačítek, textových polích pro určení první a poslední skenované adresy, textového popisku poslední dotázané adresy a grafické reprezentace postupu detekce. Tlačítko Stop slouží k pozastavení a spuštění skenování, tlačítko Reset spustí skenování od zadané první adresy po poslední. Tlačítkem Storno se formulář zavře a zbývajícími dvěma tlačítky se označené detektory v tabulce přenesou do hlavního formuláře, kde se buď do stávající tabulky detektorů přidají, nebo se vytvoří tabulka nová.



Obrázek 13: Grafické rozhraní formuláře detekce sítě.

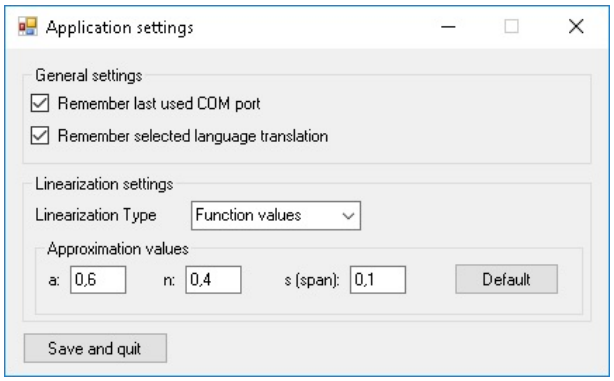
Další důležitou komponentou pro fungování programu je formulář FormPortSettings, který vyhledá v počítači dostupné sériové porty COM a dovoluje nastavit parametry sériového portu určeného ke komunikaci s detektory. Obrázek 14 znázorňuje grafické rozhraní formuláře. Ve formuláři se kromě textových polí a roletových seznamů, určených k nastavování parametrů sériového portu, nachází také tlačítko v podobě dvou šipek ve tvaru kruhu. Toto tlačítko slouží k znovu vyhledání sériový portů například v případě, kdy nemáme aktuálně žádný port dostupný

a teprve po otevření formuláře jsme připojili převodník k počítači. Tlačítkem Ok se předá zvolený port a jeho nastavení hlavnímu formuláři.



Obrázek 14: Grafické rozhraní formuláře nastavení sériového portu.

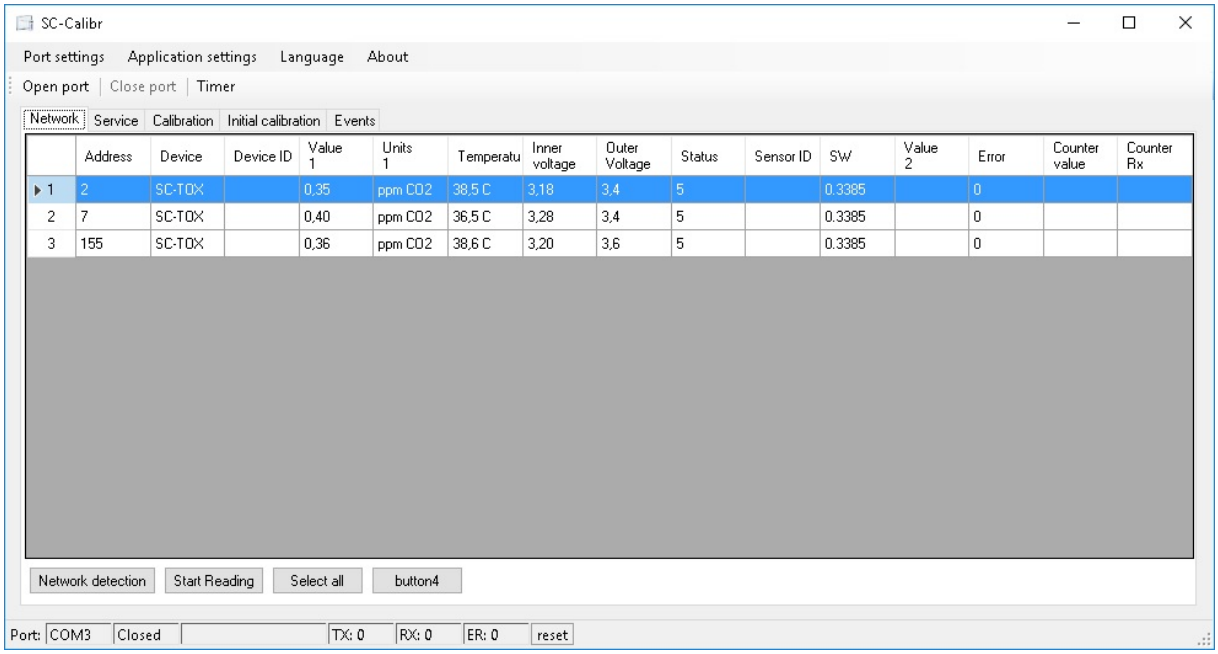
Aby nebylo nutné po každém zapnutí programu znovu nastavovat jazyk nebo sériový port, tak lze programu říct, že si má nastavení portu nebo jazyku uložit do INI souboru. Tomuto účelu slouží formulář FormApplicationSettings, jehož rozhraní je vyobrazeno ilustrací 15. Kromě nastavení výše zmíněných parametrů, lze také nastavovat typ linearizační metody a její parametry. Tlačítkem Uložit a zavřít se předají parametry nastavení hlavnímu formuláři, který požadavek zpracuje.



Obrázek 15: Grafické rozhraní formuláře nastavení programu.

Funkcionality v hlavním formuláři jsou rozděleny do jednotlivých karet, aby bylo docíleno přehlednosti programu. Kdyby program obsahoval pro každou funkci zvlášť formulář, došlo by ke ztrátě orientace uživatele. Ve formuláři se nachází hlavní nabídka obsahující položky pro vyvolání formulářů pro nastavení sériového portu, nastavení aplikace, zobrazení informací o programu, a volbu jazyka programu. Dále ve formuláři je nástrojová a stavová lišta. Nástrojová lišta má prvky pro obsluhu sériového portu (zavření a otevření) a vyvolání okna časomíry a kalkulačky koncentrace. Ve stavové liště se vypisuje zvolený sériový port, stav komunikace, typ vrácené odpovědi a počítadla odeslaných, přijatých a chybových zpráv, a také tlačítko jejich vynulování.

Doposud zmíněné prvky jsou neměnné během změny aktivní karty. Formulář je rozdělen pěti kartami, čtyři karty oddělují ovládání jednotlivých funkcionalit programu a poslední pátá karta obsahuje záznam aplikace, ze kterého lze diagnostikovat problémy programu nebo vyčíst historii vykonaných příkazů. První karta Síť, vyobrazena na 16, reprezentuje tabulku detektorů na sběrnici RS485 a obsahuje ovládací prvky pro vyvolání formuláře detekce zařízení na sběrnici, spuštění vyčítání hodnot ze zařízení a označení všech položek v tabulce.

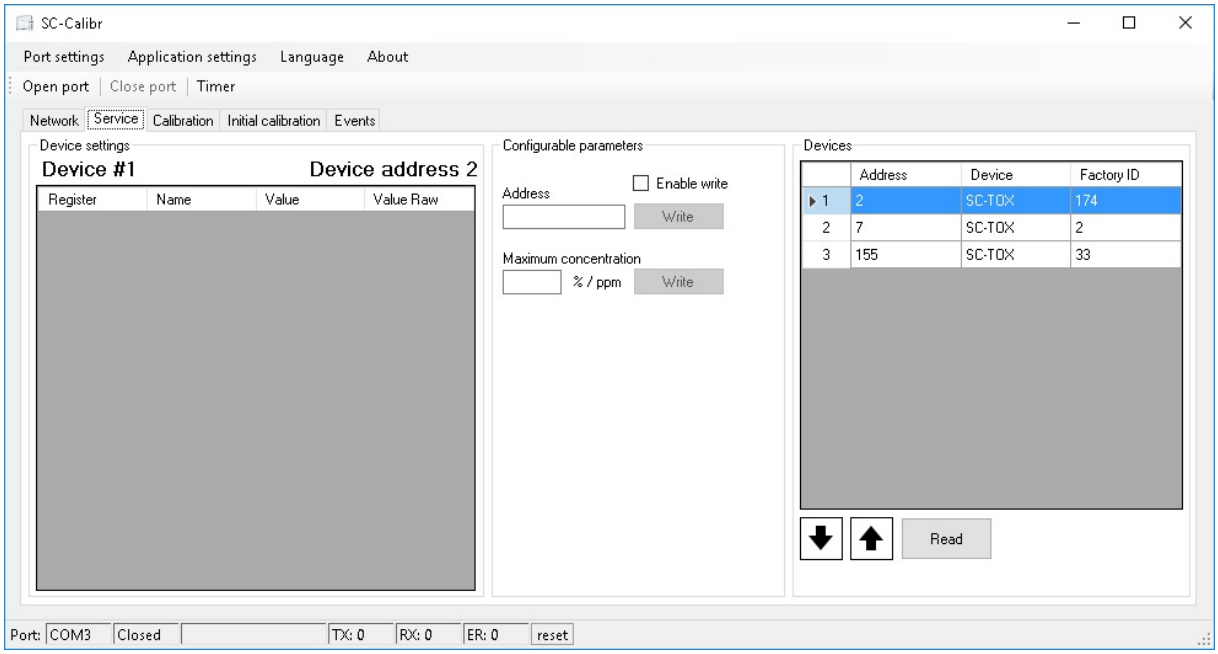


Obrázek 16: Karta zařízení v sítí.

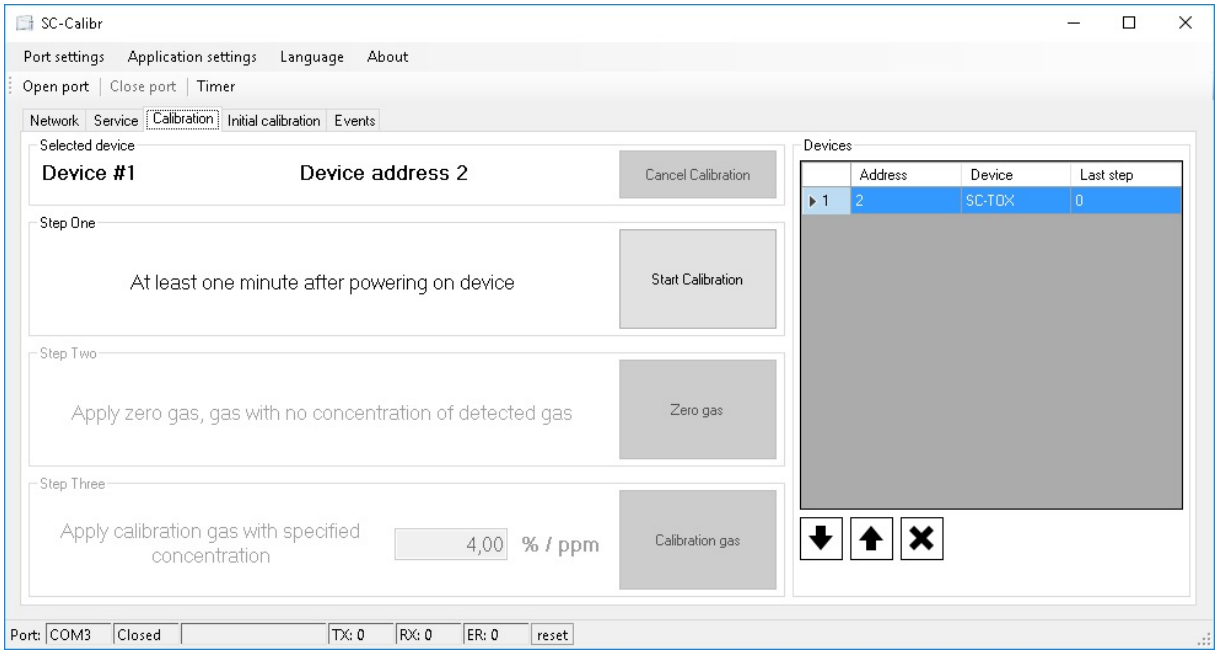
Následující karta v pořadí je určena pro komponenty nutné pro vyčítaní hodnot servisních registrů detektorů plynů a nastavování vybraných servisních registrů, které může pracovník běžně měnit. Karta je rozdělena na tři části rámečky, první obsahuje popisek pořadí a adresu aktuálně vybraného detektoru a tabulku pro vyčtené servisní registry, ve druhém rámečku se nacházejí prvky pro zápis servisních registrů, třetí rámeček má tabulku s dostupnými detektory na síti, ovládací prvky pro posun v tabulce a tlačítko pro vyčtení registrů. Uživatelské rozhraní je vyobrazeno na obrázku 17

Třetí karta seskupuje ovládací prvky pro obsluhu procesu kalibrace detektorů. Vzhled karty na ilustraci 18. Rámeček Zařízení obsahuje vybrané detektory z první tabulky v kartě Síť a ovládací prvky pro posun v tabulce a odstranění zařízení z tabulky. V rámečku Vybrané zařízení jsou popisky pořadí a adresy aktuálně zvoleného detektoru z tabulky a tlačítko pro zrušení kalibrační procedury. Další tři rámečky oddělují ovládací prvky pro jednotlivé kroky kalibrační procedury.

Poslední karta s ovládacími prvky je karta Prvotní kalibrace. Tato karta je určena výhradně pro prvotní kalibraci detektorů plynů s infračerveným senzorem, u kterých se nejdříve musí nelineární výstup senzoru převést na lineární. Karta je opět rozdělena třemi rámečky a vyobrazena



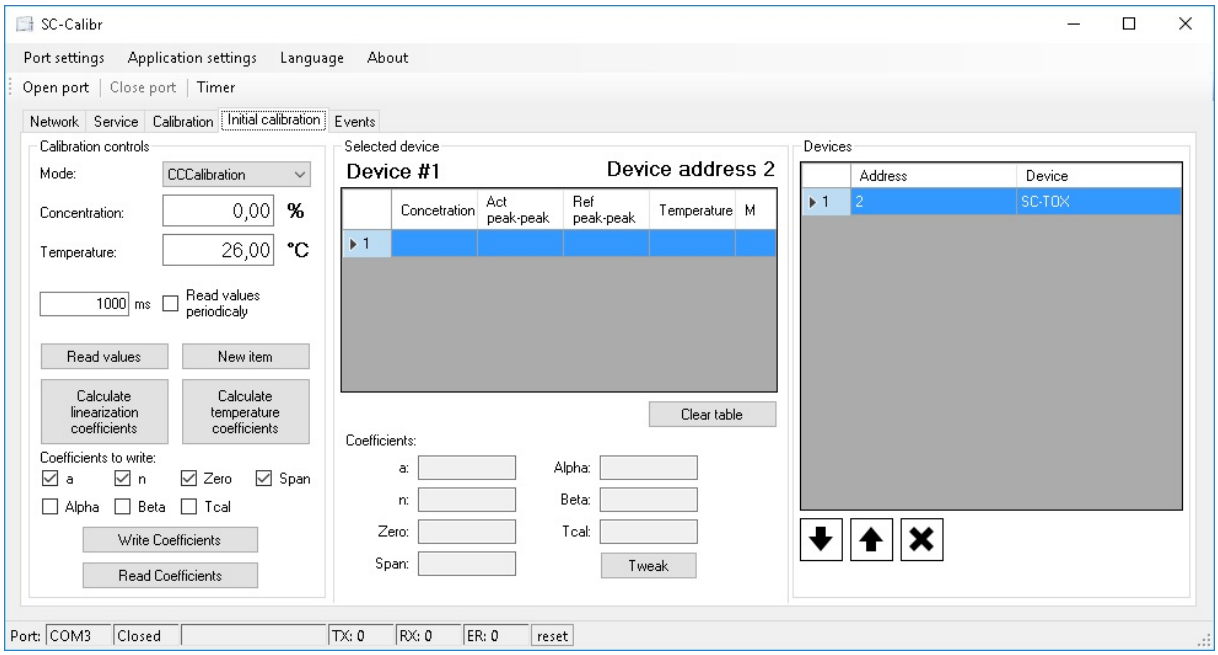
Obrázek 17: Karta servisních hodnot zařízení.



Obrázek 18: Karta kalibrace.

na obrázku 19. Pravý rámeček obsahuje tabulku kalibrovaných detektorů a ovládací prvky posunu v tabulce a odstranění detektoru z tabulky. Prostřední rámeček je tvořen popisky pořadí a adresy aktuálně zvoleného detektoru, tabulkou provedených měření a textovými poli linearizačních a kompenzačních koeficientů. Levý rámeček je určen pro ovládací komponenty kalibračních měření s textovými poli pro hodnotu koncentrace, teploty a periody pro automatické vyčítání

hodnot.



Obrázek 19: Karta počáteční kalibrace.

Formuláře FormTimerCalc a FormAbout nejsou v textové práci vyobrazeny, jelikož FormAbout obsahuje pouze popis programu a FormTimerCalc obsahuje jednoduchou kalkulačku koncentrací (kapitola 2.2.2) a časomíru.

5 Implementace programu

5.1 C#

Jedná se o vysokoúrovňový objektově orientovaný programovací jazyk založený na jazyce C++ a jazyce Java. Jazyk byl vyvinut firmou Microsoft zároveň s platformou .Net Framework a schválen standardizačními komisemi ECMA a ISO. C# lze využít k tvorbě široké škály aplikací, například databázových programů, formulářových aplikací pro operační systémy Windows, webových aplikací a stránek. Stejně jako jazyky C++ a Java je C# case-sensitive (významově rozlišuje malá a velká písmena, slovo promenna není to samé co Promenna, tato slova jsou brána jako dvě rozdílná slova). V jazyce neexistují globální proměnné a metody. Požadavkem pro běh aplikací toho jazyka je prostředí .Net Framework. Další základní charakteristiky jazyka jsou: [35]

- obsahuje nativní podporu komponentového programování,
- jednoduchá dědičnost a možnost vícenásobné implementace rozhraní,
- události,
- garbage collector se stará o uvolňování zdrojů (automatická správa paměti),
- zajišťuje typovou bezpečnost,
- podporuje zpracování chyb a výjimek.

5.2 Mono

Mono je open source projekt, vedený společnosti Xamarin, zaměřený na vývoj implementace .Net Framework společnosti Microsoft založené na ECMA standardech pro C# a Common Language Runtime. Logo 20 (obrázek převzat z článku [34]). projektu představuje opičí hlavu, jelikož slovo mono znamená ve španělštině opici. Účelem projektu Mono není pouze možnost běhu .Net aplikací na různých platformách, ale také poskytnout lepší vývojové nástroje pro vývojáře v systému Linux. Mono dokáže fungovat na různých platformách včetně Androidu, většině Linuxových distribucí, BSD, OS X, Windows, Solaris a dokonce také na některých herních konzolích jako PlayStation 3 a Xbox 360 [34].

Projekt byl v počátcích kontroverzní mezi open source komunitou, jelikož implementuje části .Net Framework, které mohou být součásti patentů firmy Microsoft. Ačkoliv standardizované části .Net Framework spadají pod závazek, že Microsoft nebude prosazovat své patenty, proti implementaci jejich specifikací pod jistými podmínkami. Kdežto jiné části .Net Framework pod tento závazek nespadají, což vedlo k obavám, že by se projekt Mono mohl stát terčem soudních procesů týkajících se porušení patentů [34].

Miguel de Icaza z firmy Ximian věřil, že .Net Framework má potenciál zvýšit produktivitu programátora a začal zkoumat, zda by byla Linuxová odnož realizovatelná. Jelikož později zjistili,

Obrázek 20: Logo projektu Mono.

že jejich malý tým nedokáže vytvořit a podporovat celý projekt, tak došlo v roce 2001 k založení open source projektu Mono. Po třech rocích vývoje byla vydána v roce 2004 první verze Mono 1.0. Mono se vyvinulo z původního záměru vývojářské platformy pro Linux na platformu podporující velkou škálu architektur a operačních systému včetně embedded systémů. Aktuální verze projektu Mono je 4.2.2. Tato verze poskytuje jádro API .Net Framework, podporu Visual Basic.Net, C# verze 2.0, 3.0, 4.0 LINQ, XML a SQL. C# verze 6.0 je výchozím nastavení C# kompilátoru. Podporovány jsou taktéž Windows Forms 2.0, ale nejsou aktivně vyvíjeny a jejich podpora je v Mono nekompletní. Verze 4.0 byla první verzí, která zahrnula originální zdrojový kód .Net Framwork vydaný firmou Microsoft. Záměr projektu Mono je docílit plné kompatibility v .Net 4.5 kromě Windows Presentation Foundation, Windows Workflow Foundation a Windows Communication Foundation. Některé chybějící části .Net Framework jsou ve vývoji v experimentálním podprojektu Mono nazvaného Olive [34].

Mono se skládá z následujících třech komponent:

- komponenty jádra,
- Mono/Linux/GNOME vývojový balík,
- balík kompatibility Microsoft.

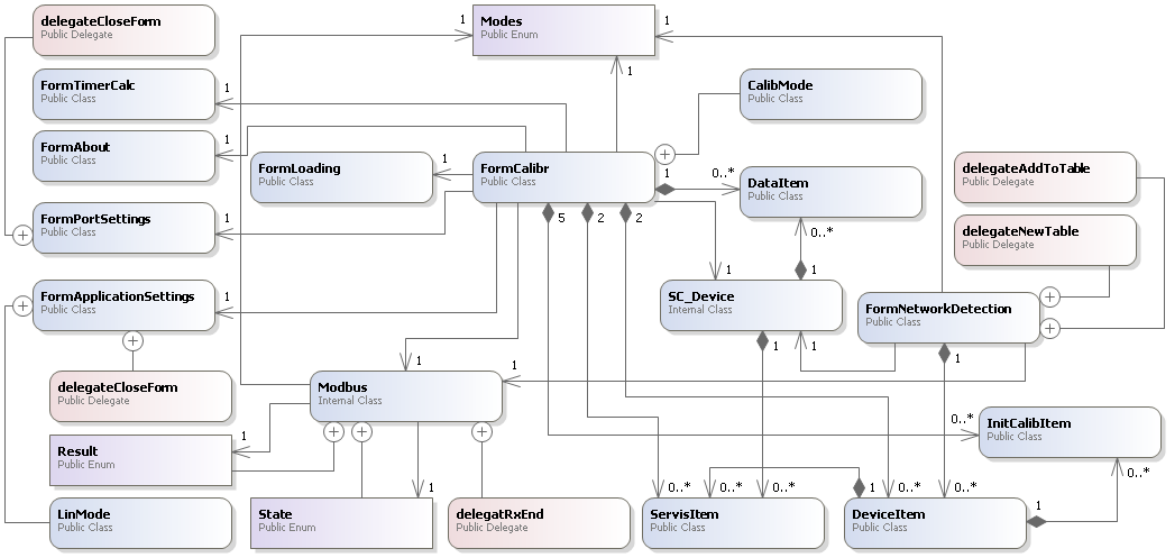
Komponenty jádra obsahují C# kompilátor, virtuální stroj pro Common Language Infrastructure a třídy knihoven jádra. Tyto komponenty jsou založeny na Ecma-334 a Ecma-335 standardech dovolujících aby Mono mohlo poskytnout bezplatný, open source virtuální stroj splňující standardy.

Mono/Linux/GNOME vývojový balík poskytuje nástroje pro vývoj aplikací s použitím existujích GNOME, bezplatných a open source knihoven. Tyto zahrnují: Gtk# pro vývoj GUI, knihovny Mozilla pro práci s rendrovacím enginem Gecko, knihovny integrace Unixu (Mono.Posix), knihovny pro databázové systémy a mnoho dalších.

Balík kompatibility Microsoft poskytuje cestu pro portování Windows .Net aplikací na GNU-/Linux. Tento balík obsahuje ADO.NET, ASP.NET, Windows Forms a další [34].

5.3 Popis komponent a implementace programu

Vztahy mezi jednotlivými komponentami programu jsou znázorněny třídním diagramem (obrázek 21). Diagram z důvodu zachování přehlednosti neobsahuje popis atributů, metod a vazeb tříd, a sdílené knihovny DLL. V této podkapitole postupně vysvětluji funkci a důvod všech použitých tříd a komponent v programu, k některým je uvedena ukázka implementace důležité části kódu. FormNetworkDetection, formulář skenuje dostupná zařízení na sběrnici a zobrazuje



Obrázek 21: Třídní diagram programu.

je v tabulce. Princip skenování je znázorněn kódem 1. Při každém tiku časovače dojde ke kontrole, zda je otevřen sériový port, poslední dotazovaná adresa (PDA) se rovná koncové adrese (PA). Pokud je poslední podmínka vyhodnocena jako platná, dojde k zastavení časovače a změně textu tlačítka, pokud platná není, tak se inkrementuje poslední dotazovaná adresa, iteruje se stav progressBaru a odešle se zpráva na sériovou linku. Tento proces se opakuje dokud není přerušen časovač tlačítkem nebo nedojde k dosažení koncové adresy, případně k uzavření portu. Formulář také obsahuje dvě vlastní události určující, jak mají být zpracována data z tabulky. Příslušná událost je vyvolána kliknutím na dané tlačítko. Implementace událostí a jejich delegátů je uvedena v kódu 2.

```
void time_Tick(object sender, EventArgs e)
{
    if (WasPortOpened) DetectDevice();
   else { time.Stop(); time.Enabled = false; buttonStop.Text = "START"; }
}

private void DetectDevice()
{
```

```
    if (LastAddress == (Stop)) { time.Stop(); time.Enabled = false; buttonStop.Text = "START"; }
    LastAddress++;
    progressBar1.PerformStep();
    modbus.Send((byte)LastAddress, (byte)03, (ushort)32768, (ushort)8, reg);
}

void modbus_RxEndEvent(int deskriptor, Modbus.Result result)
{
    int d = deskriptor;
    Modbus.Result res = result;
    nt len = modbus.indexADU;

   if (res == Modbus.Result.RX_OK)
    {
        List<DataItem> registryData = new List<DataItem>();
        List<UInt16> vyctene = new List<UInt16>();
        vyctene = modbus.readRxRegisters();
        registryData = scDevice.convertDatRegister(vyctene);

        this.Invoke((MethodInvoker)delegate()
        {
            DeviceList.Add(new DeviceItem(registryData, LastAddress));
            BindSrc.ResetBindings(false);
        });
    }
}
```

Výpis 1: Algoritmus skenování sběrnice.

```
public delegate void delegateAddToTable(List<DeviceItem> DeviceItems);
public event delegateAddToTable AddToTable;

public delegate void delegateNewTable(List<DeviceItem> DeviceItems);
public event delegateNewTable NewTable;

private void OnAddToTable(List<DeviceItem> DeviceItems)
{
   if (AddToTable != null) AddToTable(DeviceItems);
}

private void OnNewTable(List<DeviceItem> DeviceItems)
{
    if (NewTable != null) NewTable(DeviceItems);
}
```

Výpis 2: Implementace událostí a delegátů formuláře.

FormPortSettings, úkolem formuláře je vyhledat dostupné sériové porty a umožnit uživateli nakonfigurovat jeho parametry. Informace o vybraném portu a parametrech jsou předány, po kliknutí na tlačítko, událostí hlavnímu formuláři. Tlačítkem v podobě dvou šipek v kruhu lze vyčíst dostupné porty v počítači. V metodě kliknutí na tlačítko se do proměnné načtou dostupné sériové porty pomocí metody GetPortNames třídy SerialPort. Poté se do datového zdroje roletového seznamu přiřadí proměnná s vyčtenými porty. Tento algoritmus je naznačen ve výpisu 3.

```
private void buttonRefreshPorts_Click(object sender, EventArgs e)
{
    var ports = System.IO.Ports.SerialPort.GetPortNames();
    comboBoxPort.DataSource = ports;
    if (comboBoxPort.Items.Count !=0) comboBoxPort.SelectedIndex = 0;
}
```

Výpis 3: Vyčtení dostupných sériových portů v počítači.

FormAbout obsahuje pouze komponentu richTextBox s textem o programu a jeho verzi.

FormLoading zobrazuje a iteruje v časovači pouze progressBar. Formulář je určen k vizualizaci vyčítání dat ze detektorů plynů.

FormTimerCalc vykonává funkci kalkulačky procentuálních koncentrací plynu a obsahuje ovladatelný časovač pro měření časů T90, T63, T50 a T10.

FormApplicationSettings poskytuje uživatelské rozhraní pro volby uložení konfiguračních parametrů programu a nastavení linearizace v programu. Tak jako v předchozích případech je při potvrzení uložení vyvolána událost, která předá hlavnímu formuláři potřebné parametry.

Třída DeviceItem reprezentuje jeden detektor plynu. Atributy třídy jsou určeny pro uložení základních informací o detektoru vyčítaných při skenování sběrnice nebo periodickém vyčítání stavu a měřených veličin z detektorů. Třída také obsahuje pomocné atributy a seznamy pro následné interakce a funkcionality v programu.

Pro uchování vyčtených hodnot ze zařízení při měřeních během prvotní kalibrace slouží třída InitCalibItem. Třída také obsahuje pomocné atributy potřebné k fungování programu.

Konfigurační (servisní) registry z detektorů se v programu reprezentují třídou ServisItem, kdy jedna instance se rovná právě jednomu vyčtenému registru.

Podobně jako pro konfigurační registr je třída ServisItem, tak pro uchovávající registry stavu a neměřených veličin je implementována třída DataItem. Jednoduchá struktura uchovává popis a hodnotu tohoto druhu registrů.

Jelikož hodnoty z registrů jsou pouze číselné hodnoty, které jsou poté mapovány na hodnoty číselné (přepočty) i textové. K tomuto mapování slouží třída SC-Device, která provádí toto mapování a další pomocné funkce.

Výčtová třída Modes obsahuje typy komunikace protokolu MODBUS.

První myšlenkou bylo vyhledat a použít nějakou již existující knihovnu pro komunikaci protokolem MODBUS. Po prozkoumání dostupných možností jsem se rozhodl pro vlastní implementaci z důvodů především licenčních a kompatibility. Třída MODBUS obaluje veškerou obsluhu sériového portu, sestavování, odesílání a příjem zpráv protokolu MODBUS. Výpisem 4 je znázorněno sestavení části paketu protokolu MODBUS pro funkci číslo 03. Pomocí události RxEndEvent se signalizuje prvku, který je přihlášen k odběru události, že byl ukončen příjem a vrací mu stav komunikace. Vyčtená data se poté pomocí funkce readRxRegisters převedou do formátu registrů MODBUS protokolu a předají prvku v návratové hodnotě. Známým problémem na sběrnici RS485 a protokolu MODBUS je možnost kolize zpráv v případě, že na sběrnici jsou přítomny dvě a více zařízení se stejnou protokolovou adresou. Při odpovědi odpoví všechna zařízení, která odposlechla zprávu se svou adresou, a dojde ke kolizi z důvodu částečného nebo úplného překrytí zpráv. Tuto kolizi není možné detekovat běžnými prostředky. Třída tento problém částečně řeší implementováním časového limitu, do kterého musí dorazit odpověď, v opačném případě je komunikace považována za neúspěšnou. Z tohoto důvodu se předpokládá používání programu vyškolenou obsluhou, která dokáže tuto situaci logicky vydedukovat a vyřešit.

```
 public int Send(byte SlaveID, byte Function, UInt16 Address, UInt16 Quantity, UInt16[] registr)
{
    int len = 6, i, x;
    byte[] data = new byte[300];
    byte lrc;
    UInt16 crc = 0;

    data[0] = SlaveID;
    ExpectedLengthRx = 0;
    data[1] = Function;
    data[2] = (byte)(Address >> 8);
    data[3] = (byte)(Address & 0xFF);
    switch (Function)
    {
        case 3:
        {
            data[4] = (byte)(Quantity >> 8);
            data[5] = (byte)(Quantity & 0xFF);
            len = 6;
            ExpectedLengthRx = 3 + 2 * Quantity;
            ExpectedLengthRxError = 3;
            break;
        }
```

Výpis 4: Vyčtení dostupných sériových portů v počítači.

FormCalibr, komponenta, která zpracovává vše důležité pro vykonání kalibračních procedur, reprezentaci dat, výpočty linearizace a kompenzace. Shrnu zde části kódu, které jsou podle mne nejvýznamnější, nebo nějakým způsobem zajímavé. Periodické vyčítání stavu a měřených veličin z detektorů se aktivuje kliknutím na tlačítko Spustit vyčítání, které povolí časovač. Všechny použité časovače sdílejí stejnou metodu pro zpracováni události. V metodě rozlišuji časovače podle atributu Tag. Tikne-li časovač pro vyčítání hodnot, ověřím zda poslední čtený index (řádek) tabulky není větší nebo roven jak celkový počet položek seznamu (počet položek v seznamu a tabulce se shodují). Pokud je podmínka pravda, tak nastavím index na hodnotu -1 (tabulka i seznam se čísluje od nuly). Poté dojde ke inkrementaci indexu, nastavení příznaku typu odesílané potažmo očekávané zprávy, odešlu zprávu, inkrementuji počítadlo odeslaných zpráv a nechám vyznačit řádek podle aktuálního indexu. Algoritmus je uveden na výpisu 5.

```
switch(s.Tag.ToString())
{
    case "NetworkRead":
        if (LastReadIndex >= DeviceTableList.Count - 1) LastReadIndex = -1;
        LastReadIndex++;
        Expecting = 1;
        modbus.Send((byte)(int.Parse(DeviceTableList[LastReadIndex].Address)), (byte)03, (ushort)
            32768, (ushort)8, reg);
        TxCount++;
        dataGridView1.ClearSelection();
        dataGridView1.Rows[LastReadIndex].Cells[0].Selected = true;
        dataGridView1.Rows[LastReadIndex].Selected = true;
        break;
```

Výpis 5: Periodické vyčítání stavu a měřených veličin.

Pokud došlo k úspěšnému přijetí odpovědi, tak vyčtu data ze zprávy, pomocí metody třídy SC-Device namapuji hodnoty registrů na čitelné hodnoty, které uložím do seznamu. Následně obnovím tabulku (signalizuji tabulce, že došlo ke změně hodnot na které je navázaná), vyznačím řádek podle aktuálního indexu, jelikož při obnovení tabulky dojde k jejímu překreslení (vyznačí se vždy první řádek) a nastavím příznak očekávané zprávy na hodnotu nula (nic neočekávám, lze komunikovat). Výpis 6 znázorňuje kód tohoto zpracování.

```
if (res == Modbus.Result.RX_OK)
{
    RxCount++;
    List<DataItem> registryData = new List<DataItem>();
   List<UInt16> vyctene = new List<UInt16>();
    vyctene = modbus.readRxRegisters();
   switch (Expecting)
    {
        case 1:
```

```
registryData = scDevice.convertDatRegister(vyctene);
DeviceTableList[LastReadIndex] = new DeviceItem(registryData, int.Parse(
    DeviceTableList[LastReadIndex].Address));
refreshGridView(dataGridView1, null);
dataGridView1.ClearSelection();
dataGridView1.Rows[LastReadIndex].Cells[0].Selected = true;
dataGridView1.Rows[LastReadIndex].Selected = true;
Expecting = 0;
break;
```

Výpis 6: Periodické vyčítání stavu a měřených veličin.

Pro přenos dat detektorů mezi jednotlivými kartami jsem vytvořil metodu volanou při události změny aktivní karty. Kód 7 znázorňuje část metody při volbě 4. karty (karty se indexují od nuly). Nejdříve se podle označených řádků tabulky první karty vloží data do seznamu kalibrovaných zařízení, dále se navážou nové datové vazby pro tabulky ve čtvrté kartě a zavolá se metoda vypisující pořadí a adresu aktuálně vybraného zařízení.

```
var tab = sender as TabControl;
var rows = dataGridView1.SelectedRows;
var devices = new List<DeviceItem>();
foreach (DataGridViewRow r in rows)
{
    var index = r.HeaderCell.RowIndex;
    devices.Add(DeviceTableList[index]);
}
CalibDeviceList = new List<DeviceItem>(devices);
switch (tab.SelectedIndex)
{
    case 3:
        bsInitCalibDevice = new BindingSource() { DataSource = CalibDeviceList };
        dataGridViewInitialCalibDevices.DataSource = bsInitCalibDevice;
        bsInitCalibItems = new BindingSource() { DataSource = CalibDeviceList[ActiveDeviceIndex].
            ListCalibrationItems };
        dataGridViewInitialCalibDevice.DataSource = bsInitCalibItems;
        InitCalibSelectedDeviceChange();
        break;
```

Výpis 7: Algoritmus při změně aktivní karty.

Při zaslání požadavku o výpis konfiguračních registru a následném přijetí validní odpovědi jsou data zpracovávaná algoritmem 8, jedná se o pokračování kódu 6. Opět si vyčtu index označeného řádku v tabulce, namapuji vyčtené hodnoty metodou convertConfigRegisters třídy SC-Device, vyprázdním seznam přeložených registrů a nové hodnoty uložím do něj a také k příslušnému záznamu v seznamu zařízení. Do textových polí přiřadím potřebné hodnoty, obnovím

tabulku, nastavím příznak na nulu a uvolním instanci formuláře FormLoading, který indikuje vyčítání dat.

```
case 2:
    var pos = dataGridViewServisDevices.SelectedRows[0].HeaderCell.RowIndex;
    servis = scDevice.convertConfigRegisters(vyctene, uint.Parse(DeviceTableList[pos].DeviceID), int.
        Parse(DeviceTableList[pos].SenzorID));
    ServisItemsList.Clear(); ServisItemsList.AddRange(servis);
    DeviceTableList[pos].ConfigurationRegisters.Clear();//test
    DeviceTableList[pos].ConfigurationRegisters.AddRange(servis);//test
    textBoxAddress.Text = DeviceTableList[pos].Address;
    textBoxMez.Text = servis.Single(s => s.Name == "cfgMaxKonc").Value;
    refreshGridView(dataGridViewServis, null);
    Expecting = 0;
    formloading.Dispose();
    break;
```

Výpis 8: Zpracování přijatých konfiguračních registrů.

Během prvotní kalibrace je možné vyčítat měřené hodnoty automaticky v časovém intervalu. Toto je zejména vhodné, když potřebuje pracovník sledovat, kdy se měřené hodnoty ustálí. Ruční vyčítání se provádí kliknutím na tlačítko Vyčti hodnoty, algoritmus 9. Automatické vyčítání se aktivuje zaškrtnutím políčka Vyčítat hodnoty periodicky, toto políčko má na událost změny zaškrtnutí připojenou funkci, která povolí nebo zakáže časovač. Časovač při tiku zavolá metodu kliknutí tlačítka a tím dojde k vyčtení hodnot.

```
if (WasPortOpened && Expecting == 0)
{
    double c;
    var dev = CalibDeviceList[ActiveDeviceIndex];
    if (dev.Device.Contains("IR") )
    {
        if (double.TryParse(textBoxConcentration.Text, out c))
        {
            ushort regAddr = (ushort)37888;
            Expecting = 7;
            reg[0] = 0;
            LogEvent("Init calib req.: dev adr:" + dev.Address + " reg adr: " + regAddr.ToString()
                + " reg: " + reg[0].ToString());
            modbus.Send((byte)(int.Parse(dev.Address)), (byte)03, regAddr, (ushort)6, reg);
            TxCount++;
        }
    }
    else
    {
```

```
            MessageBox.Show(OnlyForIR);
        }
}
```

Výpis 9: Algoritmus vyčtení hodnot pro prvotní kalibraci.

Zápis linearizačních a kompenzačních koeficientů je volitelný, uživatel může zvolit jaké koeficienty chce zapsat. Vzhledem k této volbě se musí koeficienty zapisovat postupně. Po kliknutí na tlačítko dojde k naplnění pole logickými hodnotami podle zvolených koeficientů, vynulování proměnné koeficient k zapsání a odešle se požadavek na zápis koeficientů. Tento algoritmus je zachycen na výpisu 10.

```
WriteCoef[0] = checkBoxA.Checked; // A
WriteCoef[1] = checkBoxN.Checked; // N
WriteCoef[2] = checkBoxAlpha.Checked;// Alpha
WriteCoef[3] = checkBoxBeta.Checked;// Beta
WriteCoef[4] = checkBoxZero.Checked;// zero
WriteCoef[5] = checkBoxSpan.Checked;// span
WriteCoef[6] = checkBoxTcal.Checked; // Tcal
CoefToBeWritten = 0;
if (WasPortOpened && Expecting == 0)
{
    var dev = CalibDeviceList[ActiveDeviceIndex];
    if (dev.Device.Contains("IR"))
    {
        ushort regAddr = (ushort)37632;
        Expecting = 10;
        reg[0] = 45847; //B317
        LogEvent("Coef write req.: dev adr:" + dev.Address + " reg adr: " + regAddr.ToString() + "
            reg: " + reg[0].ToString());
        modbus.Send((byte)(int.Parse(dev.Address)), (byte)06, regAddr, (ushort)1, reg);
        TxCount++;
```

Výpis 10: Požadavek na zápis koeficientů.

Pokud dojde k přijetí odpovědi, spustí se časovač a v jeho tiku se zavolá metoda, která vždy zapíše jeden koeficient do zařízení. Postup algoritmu je zapsán výpisem 11 s ukázkou zápisu koeficientu „a“, kde dojde ke kontrole zda je sériový port otevřen a index koeficientu k zapsání je menší jak 7. Pokud hodnota pole na indexu koeficientu k zápisu je nepravdivá, tak dojde k inkrementaci indexu a znovu zavolání metody. Je-li hodnota pravdivá provede se zápis koeficientu dle indexu do zařízení.

```
private void WriteCoeficientsOneByOne()
{
    if (CoefToBeWritten < 7 && WasPortOpened)
    {
```

```
if (WriteCoef[CoefToBeWritten] == false)
{
            CoefToBeWritten++;
            WriteCoeficientsOneByOne();
}
else
{
    var dev = CalibDeviceList[ActiveDeviceIndex];
    ushort regAddr = 0;
    ushort regval = 0;
    ushort mag = Convert.ToUInt16((int)MagicNumber + (int)MagicIncrease);
    switch (CoefToBeWritten)
    {
        case 0:
            regAddr = (ushort)37633;
            Expecting = 11;
            regval = Convert.ToUInt16(Math.Truncate(Convert.ToDouble(textBoxA.Text) *
                19000.0));
            reg[0] = mag;
            reg[1] = regval;
            LogEvent("Coef A write .: dev adr:" + dev.Address + " reg adr: " + regAddr.
                ToString() + " reg: " + reg[0].ToString());
            modbus.Send((byte)(int.Parse(dev.Address)), (byte)16, regAddr, (ushort)2, reg);
            CoefToBeWritten++;
            TxCount++;
            break;
```

Výpis 11: Zápis koeficientů.

Při implementaci zápisu koeficientu jsem nalezl chybu v návrhu a kódu firmware detektorů. K zápisu koeficientu se využívá funkce číslo 06, což je funkce pro zápis jednoho registru, ale zde je funkce podivně využita k zápisu dvou hodnot, magického čísla a samotné hodnoty koeficientu. Magické číslo je generováno při požadavku zápisu a vráceno zpět zvýšené o danou hodnotu, jedná se jednoduchý princip ochrany před náhodným zápisem. Podle specifikace protokolu MODBUS, by takováto funkce neměla být zařízením zpracována. Než se ale nalezne důvod vzniku a řešení této chyby, nelze jiným způsobem zapsat uživatelem vybrané koeficienty. Je ovšem možný zápis všech koeficientů najednou pomocí funkce 16.

Program také obsahoval třídu IniParser, která zprostředkovávala čtení a zápis konfiguračních souborů ve formátu INI. Tuto třídu jsem byl nucen nahradit sdílenou knihovnou DLL, jelikož analýza portování programu pro běh na systému Linux nalezla nekompatibilitu používaného algoritmu. Více informací k tomuto problému je zmíněno v kapitole 5.6 zabývající se portováním aplikace na Linuxový systém.

5.4 Implementace jazykových variant a konfigurační soubor programu

Při implementaci překladu programu jsem nejdříve musel zvolit vhodný způsob uložení jazykových textů. Nabízelo se mnoho variant, jak překlady uložit, například ve formátu XML, prostého textového souboru nebo lokální databáze SQLite. Všechny tři zmíněné formáty jdou bez obtíží implementovat v jazyce C#, musel jsem se tedy zaměřit na rozhodnutí dle požadavků na program. Uživatel by měl být schopný sám si vytvořit jazykový překlad bez nutnosti složitých nebo nestandardních nástrojů. Tímto vypadla z výběru lokální databáze SQLite, jelikož k její editaci by bylo zapotřebí vytvořit nástroj pro zápis textů nebo znalost jazyka SQL uživatelem. Zbyly tedy textové souboru ve formátu XML a prostého textu. Formát XML má zabudovanou podporu v .Net Framework, tudíž byl ideálním kandidátem. Ovšem pro běžného uživatele může být, dle mého názoru, spleť značek velice matoucí a poměr užitného textu k části značek malý. Zvolil jsem tedy řešení za pomocí prostého textu ve formátu konfiguračního souboru INI, kde v sekci se nachází klíč a ke klíči je jeho hodnota. Soubor je poté velice přehledný, bez výplňového kódu a bezproblémový pro editaci i běžným uživatelem.

Následně popíši aktuální algoritmus 12 čtení z INI souboru za pomocí sdílené knihovny INIFileParser [27]. Vytvořil jsem si statickou metodu, která obaluje samotné čtení z INI souboru. Toto zabalení čtení do další metody se vyplatilo, jelikož při pozdějším přepisování kódu kvůli kompatibilitě s projektem Mono, jsem upravil pouze vnitřní část metody a nemusel jsem přepisovat velkou část kódu, kde potřebuji vyčíst hodnoty klíčů INI souboru. Metoda má čtyři parametry: cestu ke konfiguračnímu souboru, název sekce, název klíče a výchozí hodnotu. V metodě se vytvoří instance třídy FileIniDataParser a do objektu data se načte struktura INI souboru. Poté se dotážu na požadovaný klíč v dané sekci, pokud existuje, tak jej vrátím jako návratovou hodnotu metody. Dojde-li k chybě nebo klíč nelze nalézt vrátí metoda výchozí hodnotu zadanou parametrem funkce.

```
public static string ReadIni(string path, string Section, string Key, string DefaultValue)
{
    try
    {
        var parser = new FileIniDataParser();
        IniData data = parser.ReadFile(path);
        if (data[Section][Key] != null)
        {
            return data[Section][Key];
        }
        else
        {
            return DefaultValue;
        }
    }
    catch (Exception e)
```

```
    {
        return DefaultValue;
    }
```

Výpis 12: Čtení z INI souboru.

Aby k překladu jazykové mutace programu stačilo uživateli pouze vytvořit a zeditovat INI soubor, tak musí být v programu algoritmus 13, který automaticky prohledává složku s umístěním jazykových souborů. Nalezené soubory vloží do seznamu a zobrazí je pod položkou Jazyk v liště hlavní nabídky. Při volbě (kliknutí) jazyku dojde k zavolání metody, která zprostředkovává přeložení textů programu.

```
List<string> langs = new List<string>();
foreach (string f in Directory.GetFiles(System.IO.Path.GetDirectoryName(System.Reflection.
    Assembly.GetExecutingAssembly().Location) + "\\languages\\", "*.ini"))
{
    langs.Add(Path.GetFileNameWithoutExtension(f));
}
toolStripMenuItemLanguage.DropDownItems.Clear();
foreach (string s in langs) toolStripMenuItemLanguage.DropDownItems.Add(s);
toolStripMenuItemLanguage.ShowDropDown();
```

Výpis 13: Čtení z INI souboru.

Když už jsme měl implementováno čtení z INI souboru, tak bylo logické mít konfigurační soubor programu také v tomto formátu. Pro vyčítání hodnot z konfiguračního souboru se využívá stejná metoda jako pro čtení jazykových překladů. Zápis do konfiguračního souboru se děje pouze, když dojde k vyvolání události ve formuláři FormApplicationSettings. Kód metody, která je napojená na událost, je ve výpisu 14. Zápis je obdobný čtení, jen se zapíšou do objektu načteného INI souboru nové hodnoty klíčů a provede se zápis do souboru.

```
var path = System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().
    Location) + "\\settings.ini";
    try
    {
        var parser = new FileIniDataParser();
        IniData data = parser.ReadFile(path);
        data["Application"]["Language"] = Language;
        data["Port"]["Port"] = Port;
        data["Linearization"]["Mode"] = LinearizationMode.ToString();
        parser.WriteFile(path, data);
    }
    catch (Exception e)
    {
        LogEvent("Error while saving configuration");
    }
```

Výpis 14: Zápis do INI souboru.

5.5 Výpočty v programu

Další velice důležitou částí programu je výpočet linearizačních a kompenzačních koeficientů pro infračervené senzory. Pro ostatní typy senzorů s lineárním výstupem se přepočty koncentrací provádějí přímo v detektoru lineární rovnicí, ale vzhledem k paměťové a výpočetní náročnosti linearizace a výpočtu kompenzačních koeficientů je nutné tyto operace provádět mimo detektory. Nejdříve jsem si musel vyhledat, co vlastně musím s rovnicí udělat, abych dostal výsledné koeficienty. Po nastudování několika zdrojů jsem došel k závěru, že to, co hledám je nelineární regrese pomocí metody nejmenších čtverců. Pro výpočet linearizačních koeficientů se používá nelineární regrese (linearizace), kde vstupem jsou experimentální data $x_0 \dots x_n$ a $y_0 \dots y_n$ a matematická rovnice 1 popisující závislost experimentálních dat, kde Act, Ref jsou hodnoty vyčtené z AD převodníku detektoru plynu, C je hodnota koncentrace, hodnota $Zero$ je vypočtena rovnicí 2, $Span, a, n$ jsou hledané parametry nelineární regresí [28, 29, 30].

$$1 - \frac{Act}{Zero \times Ref} = Span \times (1 - e^{(-a \times C^n)}) \tag{1}$$

Hodnotami Act a Ref vyčtenými z AD převodníku při nulové koncentraci plynu se vypočítá rovnicí 2 hodnota $Zero$.

$$Zero = \frac{Act}{Ref} \tag{2}$$

Matematickou rovnici 1 upravíme do rovnice tvaru 3, kde x a y jsou experimentální data a $s = Span, x = C$. Tato upravená rovnice bude podrobena metodě nelineární regrese [28, 29, 30].

$$\begin{aligned} y &= 1 - \frac{Act}{Zero \times Ref}, \\ 1 - \frac{Act}{Zero \times Ref} &= Span \times (1 - e^{(-a \times C^n)}) \implies \\ y &= s \times (1 - e^{(-a \times x^n)}) \end{aligned} \tag{3}$$

Při výpočtu koeficientů teplotní kompenzace se rovnice výpočtů koeficientů α 4 a β 6 musí upravit na tvar $y = m \times x + c$, kde m je rovno α nebo β. V rovnici výpočtu α je: $NormalisedTransmittance = Act/(Zero \times Ref)$, T aktuální teplota v Kelvinech, T_{cal} teplota během kalibrační procedury [28, 29, 30].

$$NormalisedTransmittance_{(comp)} = NormalisedTransmittance \times (1 + \alpha(T - T_{cal}) \tag{4}$$

Jelikož α koeficient kompenzuje teplotu při nulové koncentraci, tak by hodnota $NormalisedTransmittance_{(comp)}$ měla být rovna jedné. Tím pádem při úpravě rovnice do tvaru $y = m \times x + c$ získáme rovnici 5 [28, 29, 30].

$$\frac{NormalisedTransmittance_{(comp)}}{NormalisedTransmittance} = \alpha \times (T - T_{cal}) + 1 \tag{5}$$

Pro získání hodnot x a y je doporučeno provést minimálně dvě měření s různými teplotami okolí s rozdílem alespoň 21 °C. Hodnotu α tedy získáme jako sklon lineární přímky [28, 29, 30].

Rovnice 6 pro výpočet koeficientu β obsahuje tyto parametry: *Span* parametr získaný během kalibrace, T aktuální teplota v Kelvinech, T_{cal} teplota během kalibrační procedury [28, 29, 30].

$$Span_{comp} = Span + (\beta \times \frac{T - T_{cal}}{T_{cal}}) \tag{6}$$

Po převodu rovnice 6 do tvaru $y = m \times x + c$ vyjde výsledná rovnice 7. Stejně jako pro koeficient α je doporučeno provést nejméně dvě měření během různých teplot okolí s rozdílem alespoň 21 °C. Jelikož se opět jedná o lineární rovnici, tak se koeficient β vypočítá jako sklon lineární přímky [28, 29, 30].

$$Span_{comp} = \beta \times \frac{T - T_{cal}}{T_{cal}} + Span \tag{7}$$

Po seznámení s postupy výpočtů jsem začal hledat, zda je již implementována třída nebo sdílená knihovna pro výpočty linearizace. Narazil jsem na velký počet různých matematicky a optimalizačně zaměřených knihoven, ale buď byly placené a jejich cena byla k tisíci dolarům nebo měly mizernou či téměř nulovou dokumentaci. Naštěstí jsem narazil na sdílenou knihovnu AlgLib [26], která existuje také kromě placené verze i v licenci GPL. Seznámil jsem se popisem výpočtů nejmenších čtverců knihovny v uživatelském manuálu a podle dokumentace jsem začal s implementací. Pro výpočet nejmenší čtverců využívá knihovna *Levenberg-Marquardt* algoritmus, což je numerická metoda pro určení minima nelineární funkce. Knihovna umožňuje provést výpočty třemi různými způsoby: [23]

- použití pouze hodnoty funkce,
- použití hodnoty funkce a gradientu,
- použití hodnoty funkce, gradientu a Hessianu.

Podle uživatelské příručky je způsob použití hodnot funkce nejpomalejší ale zároveň nejjednodušší pracovní režim. Pro výpočet stačí algoritmu dodat pouze rovnici, která se má vypočítat, vstupní hodnoty x, y a odhadované hodnoty výsledku. Při výpočtu s použitím gradientu je nutné předat algoritmu kromě předchozích parametrů také vypočítaný gradient původní funkce. Gradient je vektor obsahující jednotlivé parciální derivace funkce f v určitém bodě. Tento režim je rychlejší a může najít řešení s větší přesností než předchozí. Třetí režim zahrnuje kromě

požadavků obou předchozích režimů výpočet Hessianu (Hessovy matice) 8, výpočet by měl být vhodný pro data s velkým šumem. Hessova matice je čtvercová matice $n \times n$ obsahující všechny parciální derivace druhého řádu funkce f [23, 24, 25].

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 x_n} \\ \frac{\partial^2 f}{\partial x^2 x^1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n x_1} & \frac{\partial^2 f}{\partial x_n x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{8}$$

Rozhodl jsem se vyzkoušet implementovat všechny tři režimy a zjistit rozdíly v jejich přesnosti porovnáním. Přeci jen koeficienty je nutné vypočítat co nejpřesněji, obzvlášť jedná-li se o detekci plynů. Podle příkladu jsem naimplementoval první režim výpočtů s využitím hodnoty funkce. Naprogramování bylo bez obtíží a dokázal jsem získat koeficienty rovnice. Výsledné hodnoty jsem porovnal s hodnotami z aplikace na stránce `http://mycurvefit.com/`. Po uvážení chyby výpočtu uváděné na stránce jsem usoudil, že algoritmus pracuje správně. S nadšením jsem se vrhl na implementaci dalších dvou režimu. Musel jsem tedy nejdříve vypočítat nutný gradient a Hessian pro mojí funkci. Algoritmu se tyto parametry předávají formou statické funkce, ve které se nachází pole gradientu nebo Hessové matice. Vypočetl jsem tedy hodnoty těchto parametrů a programoval je. Výpočty gradientu funkce $y = s \ \times (1 - e^{(-a \times x^n)})$ jsou uvedeny ve vzorci 9. Sestavená Hessova matice je uvedena vzorcem 10, pro přehlednost není matice ve tvaru matice, ale jednotlivé položky jsou rozepsány na samostatný řádek pro přehlednost z důvodu délky výsledků.

$$\begin{aligned} \tfrac{\partial}{\partial s}(s \ \times (1 - e^{(-a \times x^n)})) &= 1 - e^{(-a \times x^n)} \\ \tfrac{\partial}{\partial a}(s \ \times (1 - e^{(-a \times x^n)})) &= s \ \times x^n \times e^{(-a \times x^n)} \\ \tfrac{\partial}{\partial n}(s \ \times (1 - e^{(-a \times x^n)})) &= a \times s \ \times x^n \times \log x \times e^{(-a \times x^n)} \end{aligned} \tag{9}$$

$$\begin{aligned} \tfrac{\partial^2}{\partial s^2}(s \ \times (1 - e^{(-a \times x^n)})) &= 0 \\ \tfrac{\partial}{\partial s}\tfrac{\partial}{\partial a}(s \times (1 - e^{(-a \times x^n)})) &= x^n \times e^{(-a \times x^n)} \\ \tfrac{\partial}{\partial s}\tfrac{\partial}{\partial n}(s \times (1 - e^{(-a \times x^n)})) &= a \times x^n \times \log x \times e^{(-a \times x^n)} \\ \tfrac{\partial}{\partial a}\tfrac{\partial}{\partial s}(s \times (1 - e^{(-a \times x^n)})) &= x^n \times e^{(-a \times x^n)} \\ \tfrac{\partial^2}{\partial a^2}(s \ \times (1 - e^{(-a \times x^n)})) &= s \ \times x^{(2n)} \times (-e^{(-a \times x^n)}) \\ \tfrac{\partial}{\partial a}\tfrac{\partial}{\partial n}(s \times (1 - e^{(-a \times x^n)})) &= s \ \times x^n \times logx \times (-e^{(-a \times x^n)}) \times (a \times x^{n-1}) \\ \tfrac{\partial}{\partial n}\tfrac{\partial}{\partial s}(s \times (1 - e^{(-a \times x^n)})) &= a \times x^n \times \log x \times e^{(-a \times x^n)} \\ \tfrac{\partial}{\partial n}\tfrac{\partial}{\partial a}(s \times (1 - e^{(-a \times x^n)})) &= s \ \times x^n \times \log x \times (-e^{(-a \times x^n)}) \times (a \times x^{n-1}) \\ \tfrac{\partial^2}{\partial n^2}(s \ \times (1 - e^{(-a \times x^n)})) &= -a \times s \ \times x^n \times log^2 x \times e^{(-a \times x^n)} \times (a \times x^{n-1}) \end{aligned} \tag{10}$$

Po dokončení úprav jsem spustil algoritmus pro každý režim a pokaždé jsem dostal stejné výsledné hodnoty. Nebyl jsem si zcela jistý, zda je to správně a vzhledem k tomu, že jsem se setkal s Hessovou maticí poprvé a nebyl jsem si jist, jak ji přesně přenést do formátu pole požadovaného algoritmem, tak jsem se dotázal autora knihovny ohledně správnosti matice. Autor mi odpověděl,

že zkontrolovat matici není jednoduché, ale poukázal na chybu v mém kódu a linearizaci pomocí gradientu ani Hessové matice neprovádím, jelikož mi v kódu chybí konstruktory funkce pro tyto režimy. Opravil jsem kód, ale výpočet vždy vracel pouze odhadové hodnoty. Obrátil jsem se tedy znovu na autora, který můj problém dokázal identifikovat. Problém byl relativně triviální, ale já ho nepostřehl. Výpočet se vždy zastavil jelikož v derivacích se vyskytuje $logx$ a mezi vstupními hodnotami x se nachází nula. Logaritmus hodnoty nula je roven nekonečnu, takže průběh algoritmu se zastavil. Řešením je ošetřit vstupní hodnoty, to znamená nahradit nuly velmi malým číslem. Ve výpisu kódu 15 vidíme ukázku výpočtu koeficientu v režimu použití hodnoty funkce, gradientu a Hessianu.

```
double epsf = 0;
double epsx = 0.000001;
int maxits = 0;
int info;
alglib.lsfitstate state;
alglib.lsfitreport rep;
double diffstep = 0.000001;
double[] c = new double[] { 0.05, 0.4, 0.3 };
double[] bndl = new double[] { 0.0001 , 0.0001 , 0.0001 };
double[] bndu = new double[] { 2.0 , 2.0 , 2.0 };
alglib.lsfitcreatef (x, y, c, diffstep , out state);
alglib.lsfitsetbc (state, bndl, bndu);
alglib.lsfitsetcond (state, epsf, epsx, maxits);
alglib.lsfitfit (state, function_cx_1_func, function_cx_1_grad, function_cx_1_hess, null, null);
alglib.lsfitresults (state, out info, out c, out rep);

public static void function_cx_1_func(double[] c, double[] x, ref double func, object obj)
{
    func = c[0] * (1 - Math.Exp(-c[1] * Math.Pow(x[0], c[2]))); // s*(1-exp(-a*pow(x,n)))
}

public static void function_cx_1_grad(double[] c, double[] x, ref double func, double[] grad, object
    obj)
{
    func = c[0] * (1 - Math.Exp(-c[1] * Math.Pow(x[0], c[2])));
    grad[0] = 1 - Math.Exp(-c[1] * Math.Pow(x[0],c[2]));
}

public static void function_cx_1_hess(double[] c, double[] x, ref double func, double[] grad, double
    [,] hess, object obj)
{
    hess[2, 2] = -c[1] * c[0] * Math.Pow(x[0], c[2]) * Math.Pow(Math.Log(x[0]),2) * (-Math.Exp(-
        c[1] * Math.Pow(x[0], c[2]))) * (c[1] * Math.Pow(x[0], (c[2] - 1)));
}
```

Výpis 15: Čtení z INI souboru.

Jakmile byly všechny režimy funkční, spustil jsem výpočty pro všechny režimy a různé hodnoty ošetření nuly (nulu jsem nahradil malým číslem: 0,0001 0,000001 a 0,00000001). Tabulka 29 shrnuje porovnání parametrů výpočtu a hodnot získaných koeficientů. Parametr R^2 je koeficient determinace (podíl rozptylu) v intervalu $< 0, 1 >$ a vyšší hodnoty znamenají větší úspěšnost regrese. V tabulce jsou uvedeny hodnoty, tak jak byly vráceny algoritmem, vzhledem k velmi malým rozdílům jsem se rozhodl hodnoty nijak zaokrouhlovat a uvést je tak, jak byly vráceny. Pozorováním tabulky jsem zjistil, že všechny tři režimy vracejí téměř totožné hodnoty, které se shodují na šest desetinných míst. Pro linearizační koeficienty je důležitých prvních pět desetinných míst, lze tedy říci, že pro tento výpočet jsou si režimy rovny. Pokud by se měl vybrat režim podle nejmenší chybovosti, byl by to režim za použití Hessovy matice, ten vykazuje nejmenší průměrnou chybu a prohlásil bych jej za nejpřesnější. Praktické zkušenosti později ukážou, který režim bude nakonec nejvhodnější, proto je v programu volba výběru režimu výpočtu.

Tabulka 29: Porovnání parametrů režimů algoritmu linearizace.

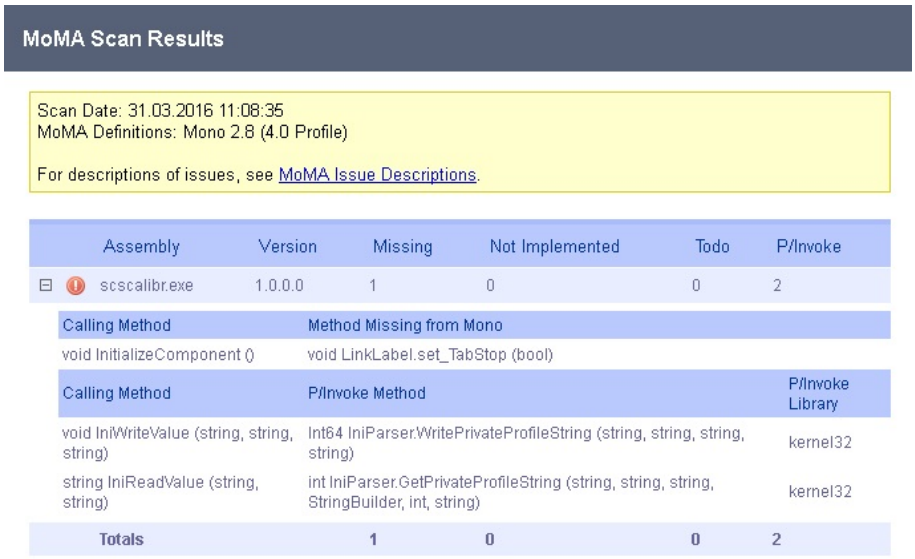
Režim	**s**	**a**	**n**	R^2	**Maximální chyba**	**Průměrná chyba**
0,0001						
hodnoty funkce	0,2573419282	0,7573883756	0,5878189292	0,9998725041	0,0013985427	0,0006317837
gradient	0,2573420048	0,7573879911	0,5878187593	0,9998725041	0,0013985334	0,0006317849
hessian	0,257341739	0,7573891997	0,587819522	0,9998725041	0,0013985578	0,000631782
0,000001						
hodnoty funkce	0,2579095595	0,7550046041	0,5859411481	0,9998903127	0,0013586219	0,0005610895
gradient	0,2579093878	0,7550053746	0,5859416386	0,9998903127	0,0013586374	0,000561088
hessian	0,2579092936	0,7550057842	0,5859419264	0,9998903127	0,001358645	0,0005610874
0,00000001						
hodnoty funkce	0,2579137161	0,7549871301	0,5859275881	0,999890395	0,0013583285	0,0005560741
gradient	0,2579135429	0,7549879072	0,5859280829	0,999890395	0,0013583441	0,0005560726
hessian	0,2579135299	0,7549879656	0,58592812	0,999890395	0,0013583453	0,0005560725

5.6 Portování pro OS Linux

Projekt Mono má přehledně zpracovanou dokumentaci a dohledat informace o portování (převodu) aplikace pro operační systém Linux nebyl problém. V dokumentaci je zmíněno, že úsilí nutné k převodu aplikace se může velice lišit projekt od projektu. Některé aplikace mohou fungovat bez modifikací, ale většina vyžaduje úpravu, aby fungoval plynule. Jako první krok může vývojář provést analyzování kódu nástrojem MoMa (Mono Migration Analyzer). Tento nástroj pomůže odhalit potenciální problémy, které mohou vzniknout při portování aplikace pro Mono. Ačkoliv se Mono zaměřuje na binární kompatibilitu s .Net Framework, tak nástroj MoMa dokáže určit platformně specifické volání (P/Invoke) a části kódu, které nejsou podporovány projektem Mono. I přes to, že nástroj může odhalit možné problémy, tak pořád existují případy, které nemohou být pokryty nástrojem. Zároveň nástroj nemusí odhalit problémy správně a vyhodnotit nezávadné části kódu jako nekompatibilní [37].

5.7 Analýza a portování kódu

Podrobil jsem výsledný program nástroji MoMa podle dokumentace a provedl analýzu. Výsledkem byla zpráva analýzy s několika problémy ohledně kompatibility, viz obrázek 22. Jelikož nástroj MoMa je pouze orientační, rozhodl jsem se zároveň vyzkoušet program na systému Ubuntu. Po spuštění programu nedošlo ani k načtení grafického rozhraní a program havaroval.



Obrázek 22: Zpráva analýzy nástroje MoMa.

Důvod havárie bylo právě volání (P/Invoke), přesněji kernel32.dll, které jsem využíval ve vlastní třídě pro práci s konfiguračním INI souborem. Kernel32.dll poskytuje aplikacím většinu API založených na Win32, jako správu paměti, I/O operace a další. Potřeboval jsem tedy najít náhradu, nebo implementovat čtení jinou cestou. Při kombinaci klíčových slov Mono a INI jsem narazil na sdílenou knihovnu pro práci s INI soubory kompatibilní s projektem Mono. [27]

Díky tomu, že jsem čtení dat z INI souboru měl obaleno další metodou ve třídě formuláře, tak změna implementace znamenala upravit pouze vnitřní kód metody. Zároveň jsem odstranil další problém identifikovaný nástrojem MoMa, a to použití odkazového popisku. Tento popisek jsem nahradil obyčejným popiskem. Zdrojový kód jsem zkompiloval a provedl další analýzu, které proběhla úspěšně bez identifikace jakýchkoliv problémů.

5.8 Běh na OS Linux

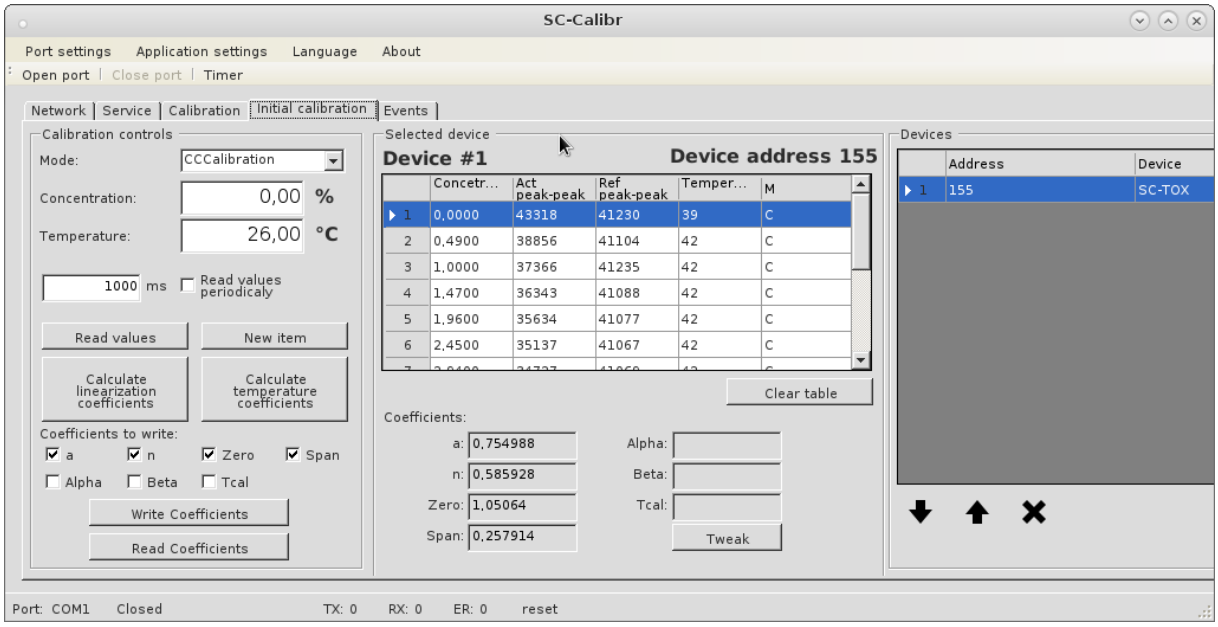
V prostředí VirtualBox verze 5.0.12 jsem si nainstaloval dva Linuxové virtuální počítače s distribucemi Debian (Jessie 8) a Ubuntu (14.04.1). Na obou strojích jsem nainstaloval Mono pomocí příkazů 16

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 3
    FA7E0328081BFF6A14DA29AA6A19B38D3D831EF
echo "deb http://download.mono-project.com/repo/debian wheezy main" | sudo tee /etc/apt/sources.
    list.d/mono-xamarin.list
sudo apt-get update
sudo apt-get install mono-complete
```

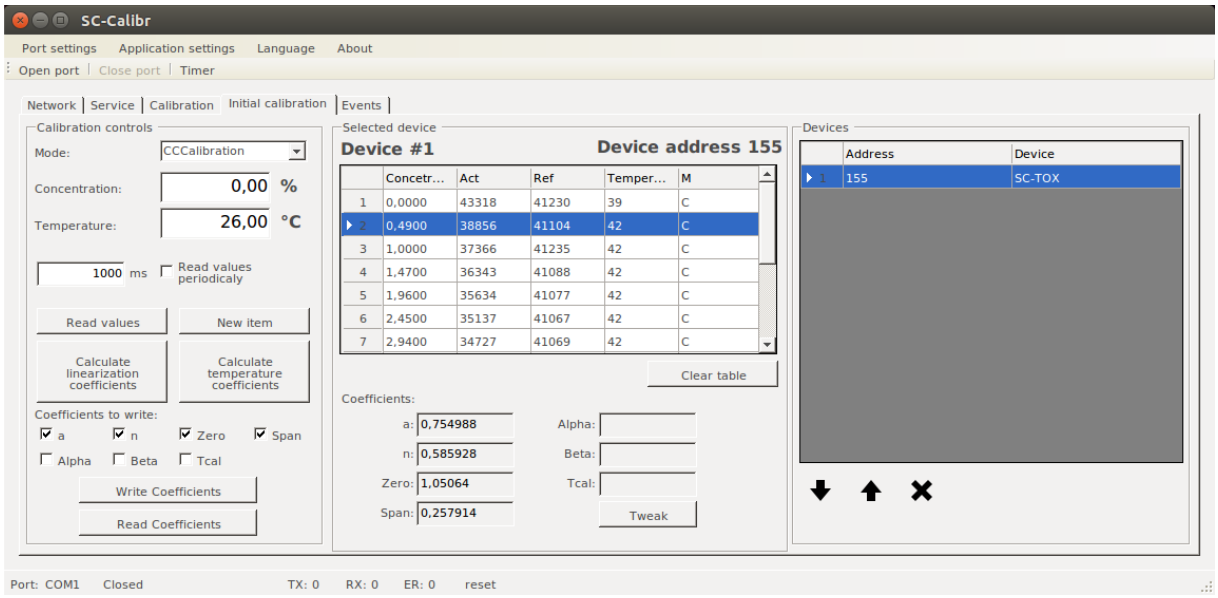
Výpis 16: Příkazy instalace projektu Mono

Program se úspěšně podařilo spustit pod oběma systémy, obrázky 23 a 24 jsou vzhled programu na systému Debian a Ubuntu. Oproti běhu na systému Windows vykazuje program mírné grafické rozdíly. Na distribuce Debian je grafický vzhled horší, především jsou rozměrově špatné tabulky, toto přisuzuji virtualizovanému systému potažmo ovladačům, jelikož nemůžu změnit rozlišení systému. Horší to je v oblasti funkčnosti programu.

Hlavním problém je obsluha sériového portu, kdy v distribuci Debian program dokáže otevřít sériový port, ale vysílaná nejspíše data neodpovídají tomu, jak by měly vypadat. Přenos na sběrnici probíhá (indikace LED na převodníku), ale zařízení neodpovídají. U distribuce Ubuntu nedojde ani k otevření sériového portu, komponenta serialPort vrátí hodnotu Open failed. Během zkoušení programu došlo párkrát také k nenadálé havárii. Vyzkoušel jsem tedy program spustit s parametrem „–debug“, zda zjistím příčinu pádů. V jednom případě došlo k havárii kvůli neplatné cestě k souborům, systém Windows používá jako oddělovací znak „/“ oproti systému Linux, který používá „\“. Dalším častým důvodem havárie byla práce s komponentou dataGridView, která nejspíše není zcela podporována nebo s ní v kódu špatně zacházím. Aby byla aplikace plně funkční v systému Linux, tak je nutné provést ladění a odzkoušení přímo v systému, což by bylo jistě velice časově náročné. Je možné, že problém komunikace je způsoben virtualizovaným systémem. Program je aktuálně nefunkční po stránce komunikace pod systémem Linux, to lze považovat za výzvu do budoucna a program odladit.



Obrázek 23: Vzhled programu v distribuci Debian.



Obrázek 24: Vzhled programu v distribuci Ubuntu.

6 Ověření funkcionalit

Firma ZAM-SERVIS s.r.o. mi zapůjčila malou tlakovou láhev naplněnou metanem s koncentrací cca 2,6 % a několik detektorů, pro otestování vyvinutého programu. Na fotografii 25 je zachycena tlaková láhev a detektory po zkušební kalibraci. Nejprve jsem otestoval vyčítání hodnot ze zařízení při vystavení detektoru plynem. Hodnoty zobrazované v okně programu se shodovaly s hodnotami zobrazovanými na displeji zařízení, jednalo se o hodnotu 2,64 %. Dále jsem otestoval nastavení servisních parametrů, kdy došlo k úspěšnému zapsání síťové adresy a meze maximální koncentrace (koncentrace, při které sepne tranzistorový výstup). Parametry jsem ověřil výpisem servisních registrů.

Samozřejmostí bylo ověření funkčnosti kalibrační procedury. Nejprve jsem vyzkoušel zkalibrovat již kalibrovaný snímač, zda bude zobrazovaná hodnota koncentrace stejná. Po dokončení kalibrační procedury byly hodnoty zobrazované koncentrace totožné, pokud vezmu v potaz kolísání koncentrace v okolí snímacího elementu. Připravil jsem si druhé zařízení, které nebylo zkalibrováno, a provedl kalibraci se zadanou hodnotou 2,64 %. Čidlo po ukončení procedury vypsalo na displeji hodnotu 2,65 %. Vyzkoušel jsem kalibraci znovu, ale zadal jsem hodnotu koncentrace 2 %. Na displeji čidla se zobrazilo 1,99 %, v tomto případě jsem „zalhal“ o koncentraci, zda zařízení tuto podvodnou hodnotu vypíše. Z výše uvedených informací tedy usuzuji, že vyčítání, zápis dat a kalibrační procedura je implementována správně a hodnoty se do zařízení zapisují tak, jak mají.

Program v porovnání s původním programovým vybavení vyčítá a zapisuje hodnoty do zařízení stejně spolehlivě a rozšiřuje funkcionality programu. Program minimalizoval počet nutných formulářů, což zpřehlednilo program a samotnou práci. Stabilita a spolehlivost běhu aplikace je zaručena díky vývoji v jazyce C#, program jsem otestoval na různých systémech od Windows XP (Servis Pack 3) po Windows 10.

Jelikož je pro vícebodovou prvotní kalibraci potřeba provést měření s minimálním počtem tří různých koncentrací plynu (čím více koncentrací, tím lépe), domluvil jsem si s pracovníkem z vývoje detektorů s infračerveným senzorem ověřovací kalibrace pro různé zařízení a plyny. Ve spolupráci s Ing. Žvakem jsme provedli kalibrace detektorů SC-IR (infračervený senzor) určené pro plyny metan (CH_4) a oxid uhličitý (CO_2). V obou případech jsme využili ředičku plynů, do které vstupoval dusík (N) ve 100 % koncentraci a oxid uhličitý (CO_2) v koncentraci cca 4,8 %. Pro detektor CO_2 jsme provedli jedenáct měření v rozmezí koncentrace 0 až 4,8 %. Tabulka 30 popisuje hodnoty ředěné koncentrace, koncentrace změřené na výstupu ředičky referenčním detektorem a naměřené hodnoty koncentrace detektorem s infračerveným senzorem po provedené vícebodové kalibraci vyvinutým programem. Podobně jsme poté postupovali u detektoru metanu, kde jsme provedli celkem šest měření v rozsahu koncentrace 0 až 2,5 %. Do ředícího systému byl přiveden 100% dusík a metan s koncentrací 2,5 %. Vícebodová kalibrace byla poté ověřena s výsledky v tabulce 31. Po vyhodnocení výsledku ověření kalibrace byl program označen Ing. Žvakem jako spolehlivý a výborný v oblasti vícebodové kalibrace a výpočtu kalibračních

Obrázek 25: Fotografie zkušební kalibrace detektorů.

(linearizačních) a teplotních koeficientů. U programu bylo zejména zhodnoceno ulehčení práce a extrémní ulehčení nutných výpočtů, program bude moci obsluhovat i naprostý laik bez znalosti vyšší matematiky. Na fotografii 26 je zachycen průběh vícebodové kalibrace, na fotografii lze vidět ředičky plynů (zelené zařízení), tlakovou láhev CO_2, kalibrované detektory a napájecí akumulátor. Mimo záběr je řídící počítač a tlaková láhev s dusíkem. Kompletní proces kalibrace trval okolo jedné hodiny.

Zároveň jsme na dílně provedli ověřovací provozní kalibrace zařízení SC-CH4 pro plyn metan v koncentraci 2,5 %, SC-TOX pro plyn oxid uhelnatý koncentrace 426 ppm a sirovodík v koncentraci 30 ppm, a SC-IR pro plyn oxid uhličitý s koncentrací 4,8 %. Při všech kalibračních procedurách došlo k úspěšnému dokončení kalibrace a zapsání hodnot do zařízení. Při ověření správnosti kalibrace všechny detektory indikovaly stejnou hodnotu koncentrace nebo hodnotu ji velmi blízkou v rámci tolerance.

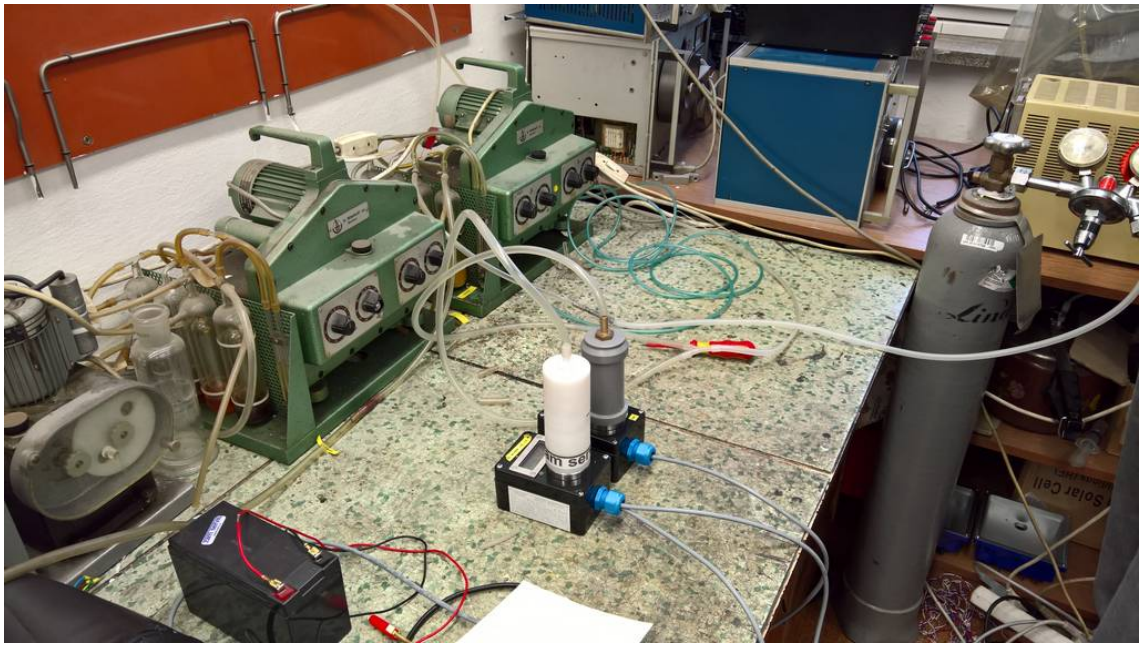
Z dosažených výsledků ověřovacích kalibrací a zkušebního čtení a zápisu dat do zařízení mohu usoudit, že vyvinuté programové ovládání zvládá požadované úkoly a vykonává je spolehlivě.

Tabulka 30: Ověřovací vícebodová kalibrace CO_2.

Ředěná koncentrace	**Koncentrace na výstupu ředičky**	**Naměřená koncentrace**
0,0 %	0,00 %	0,00 %
0,5 %	0,49 %	0,50 %
1,0 %	0,96 %	0,98 %
1,4 %	1,35 %	1,34 %
1,9 %	1,82 %	1,83 %
2,4 %	2,26 %	2,24 %
2,9 %	2,74 %	2,73 %
3,4 %	3,21 %	3,20 %
3,8 %	3,59 %	3,57 %
4,3 %	4,15 %	4,16 %
4,8 %	4,62 %	4,63 %

Tabulka 31: Ověřovací vícebodová kalibrace CH_4.

Ředěná koncentrace	**Koncentrace na výstupu ředičky**	**Naměřená koncentrace**
0,0 %	0,00 %	0,00 %
0,5 %	0,49 %	0,51 %
1,0 %	0,97 %	0,98 %
1,5 %	1,43 %	1,43 %
2,0 %	1,89 %	1,88 %
2,5 %	2,36 %	2,36 %



Obrázek 26: Fotografie zkušební vícebodové kalibrace detektorů.

7 Závěr

Výsledkem práce je funkční programové vybavení pro servis, kalibraci a výpočty kalibračních koeficientů detektorů plynů. Program umožňuje vytváření jazykových překladů uživateli velmi jednoduchým způsobem, a to vytvořením a editací prostého textového souboru. Kompatibilita se systémem Windows XP je zaručena použitím .Net Framework verze 4.0, tento předpoklad jsem ověřil spuštěním programu ve virtualizovaném systému Windows XP. Pro výpočet kalibračních koeficientů se využívá principu nejmenších čtverců s použitím Levenberg-Marquardt algoritmu, k výpočtům jsem využil sdílenou knihovnu AlgLib. Pokusil jsem se také o převedení aplikace pod operační systém Linux pomocí projektu Mono. Ačkoliv analýza portování po úpravě kódu nenalezla problémy, tak program není schopný komunikace přes sériový port, ostatní funkcionality, především výpočty fungují. Nefunkčnost pod systémem Linux je motivací pro další vývoj programu.

Provedl jsem zkoušku aplikace na funkčních detektorech, kdy jsem odzkoušel čtení a zápis ze zařízení, kalibrační proceduru a výpočty koeficientů testovací sadou dat. Otestování vícebodové kalibrační procedury a výpočty kalibračních a teplotních koeficientů bylo provedeno ve spolupráci se zaměstnancem firmy v jejím sídle. Ověřen byl jak postup a výpočty koeficientů vícebodové kalibrace pro více detektorů plynů, tak provozní kalibrace pro různé druhy zařízení a detekovaných plynů. Program byl vyhodnocen jako spolehlivý a velice užitečný.

Tato práce má pozitivní dopad na reálné použití, jelikož díky novému programovému vybavení je možné kalibrovat více detektorů zároveň, což ušetří výdaje na kalibraci a lidský čas. Výpočty kalibračních koeficientů se provádějí přímo v programu, takže není nutné využívat jiné externí nástroje jako doposud. Vyčíslení úspor financí a času bude možné zhodnotit až po delší době používání programu. Sám si netroufnu provádět jakékoliv odhady.

V tomto tématu vidím možnost dále pokračovat, jelikož aktuální stav programového vybavení na operačním systému Linux není plně funkční a chtěl bych se pokusit jej vyladit a odstranit problémy. To je časově náročný problém a proto jsem jej nezahrnul do této práce. Projekt Mono má veliký potenciál, který snad Microsoft uchopí do správných rukou.

Díky této práci jsem se dostal ke komunikaci přes sériový port a protokol MODBUS, se kterým jsem doposud nesetkal. Seznámil jsem se také funkcí LINQ, kterou jsem v práci párkrát využil při vyhledávání v seznamech datových struktur. Téma linearizace metodou nejmenších čtverců bylo velice zajímavé, ale ze začátku obtížné k pochopení a nalezení vhodné cesty implementace. Práci hodnotím ze svého pohledu pozitivně, jelikož jsem se seznámil s novými technologiemi a vyzkoušel portování programu určeného pro OS Windows na OS Linux, ke kterému bych se běžně nedostal.

Literatura

[1] ČESKÝ NORMALIZAČNÍ INSTITUT. *ČSN EN 45544-4: Ovzduší na pracovišti - Elektrické přístroje používané pro přímou detekci a přímé měření koncentrace toxických plynů a par - Část 4: Pokyny pro volbu, instalaci, použití a údržbu.* Český normalizační institut, 2001.

[2] ČESKÝ NORMALIZAČNÍ INSTITUT. *ČSN EN 60079-29-2: Výbušné atmosféry - Část 29-2: Detektory plynů - Výběr, instalace, použití a údržba detektorů hořlavých plynů a kyslíku.* Český normalizační institut, 2008.

[3] STANĚK, Jan, ŘEHÁK, Jan. RS 485 & 422. *HW.cz / Vše o elektronice a programování* [online]. 1998 [cit. 2016-03-29]. Dostupné z: http://www.hw.cz/teorie-a-praxe/dokumentace/rs-485-422.html

[4] KULICH, Martin a Martin KELTOŠ. *Problematika nebezpečí výbuchu v návaznosti na požadavky a výběr elektroinstalace a ochranných systémů* [online]. Brno, 2011 [cit. 2016-03-29]. Dostupné z: http://www.crr.vutbr.cz/system/files/brozura_06_1105.pdf

[5] *MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b* [online]. 1.1b. 2006 [cit. 2016-03-29]. Dostupné z: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf

[6] *Detekce plynů a bezpečnostní a přístrojová technika* [online]. [cit. 2016-03-29]. Dostupné z: http://people.tuke.sk/jan.kizek/bezpe/Detekcia%20plynov.pdf

[7] Provozní analyzátory plynů. *Časopis Automa* [online]. 2001 [cit. 2016-03-29]. Dostupné z: http://automa.cz/index.php?id_document=33683

[8] Pelistorové senzory – vlastnosti a použití. *Časopis Automa* [online]. 2004 [cit. 2016-03-29]. Dostupné z: http://automa.cz/pelistorove-senzory-—vlastnosti-a-pouziti-32653.html

[9] Způsoby ochrany před výbuchem – část 1: Primární protivýbuchová ochrana. *TZB-info* [online]. 2014 [cit. 2016-03-29]. Dostupné z: http://www.tzb-info.cz/pozarni-bezpecnost-staveb/10972-zpusoby-ochrany-pred-vybuchem-cast-1-primarni-protivybuchova-ochrana

[10] Přenosné analyzátory plynů. *OKD HBZS, a.s.* [online]. [cit. 2016-03-29]. Dostupné z: www.hbzs-ov.cz/dokums_soubory/prenosne_analyzatory_0.ppt

[11] Detekce a analýza plynů. *Chromservis.eu* [online]. [cit. 2016-03-29]. Dostupné z: https://www.chromservis.eu/s/gas-detection?offset=20

[12] JIRSA, Jakub. Infračervená absorpční spektroskopie a detekce plynů. *Aldebaran bulletin* [online]. 2015 [cit. 2016-03-29]. Dostupné z: http://www.aldebaran.cz/bulletin/2015_40_ird.php

[13] What is %LEL and %UEL. *RKI Instruments* [online]. [cit. 2016-03-29]. Dostupné z: http://www.rkiinstruments.com/pages/faq/What_is_LEL_UEL.htm

[14] ADÁMEK, Martin. *Obecně o elektrochemických senzorech.* [online]. [cit. 2016-03-29]. Dostupné z: http://www.umel.feec.vutbr.cz/ adamek/uceb/DATA/s_9_1.htm

[15] ADÁMEK, Martin. *Elektrochemické články.* [online]. [cit. 2016-03-29]. Dostupné z: http://www.umel.feec.vutbr.cz/ adamek/uceb/DATA/s_9_2_2.htm

[16] Gázszenzorok. *Hoeller Electronic Kft.* [online]. [cit. 2016-03-29]. Dostupné z: http://www.hoeller.hu/gazszenzorok

[17] Fundamentals of Combustible Gas Detection. *IDS-Environment* [online]. [cit. 2016-03-29]. Dostupné z: http://www.ids-environment.com/Common/Paper/Paper_101/Fundamentals of Combustible Gas Detection.htm

[18] Oxygen Sensors. *City Technology Ltd* [online]. [cit. 2016-03-29]. Dostupné z: http://www.citytech.com/technology/02-sensors.asp

[19] Toxic Sensors. *City Technology Ltd* [online]. [cit. 2016-03-29]. Dostupné z: http://www.citytech.com/technology/toxic-sensors.asp

[20] Pellistors. *City Technology Ltd* [online]. [cit. 2016-03-29]. Dostupné z: http://www.citytech.com/technology/pellistors.asp

[21] Infra-red sensors. *City Technology Ltd* [online]. [cit. 2016-03-29]. Dostupné z: http://www.citytech.com/technology/irsensors.asp

[22] RONEŠOVÁ, Andrea. *Přehled protokolu MODBUS* [online]. Plzeň, 2005 [cit. 2016-03-29]. Dostupné z: http://home.zcu.cz/ ronesova/bastl/files/modbus.pdf

[23] Least squares fitting (linear/nonlinear) - ALGLIB. *ALGLIB* [online]. [cit. 2016-03-31]. Dostupné z: http://www.alglib.net/interpolation/leastsquares.php#levmar

[24] Gradient (matematika). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-03-31]. Dostupné z: https://cs.wikipedia.org/wiki/Gradient_(matematika)

[25] Hessian matrix. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-03-31]. Dostupné z: https://en.wikipedia.org/wiki/Hessian_matrix

[26] *ALGLIB* [online]. [cit. 2016-03-31]. Dostupné z: http://www.alglib.net/

[27] *GitHub - rickyah/ini-parser* [online]. [cit. 2016-03-31]. Dostupné z: https://github.com/rickyah/ini-parser

[28] E2V TECHNOLOGIES (UK) LIMITED. *Infrared Sensor Application Note 5 Determining Coefficients for Linearisation and Temperature Compensation* [online]. 2007 [cit. 2016-03-31]. Dostupné z: http://www.e2v.com/shared/content/resources/File/sensors_datasheets/IR/infrared_an5.pdf

[29] E2V TECHNOLOGIES (UK) LIMITED. *Infrared Sensor Application Note 2 Signal Processing for Infrared Gas Sensors* [online]. 2007 [cit. 2016-03-31]. Dostupné z: http://www.e2v.com/shared/content/resources/File/sensors_datasheets/IR/infrared_an2.pdf

[30] ŽVAK, Václav. *Postup vícebodové kalibrace a výpočet kalibračních koeficientů*. ZAM-SERVIS s.r.o. 2015.

[31] ZAM-SERVIS s.r.o. *Komunikační protokol SC-CH4, SC-LCD, SC-CH4-TM, SC-TOX*. 2015.

[32] MODBUS. *In: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2016-03-29]. Dostupné z: https://cs.wikipedia.org/wiki/Modbus.

[33] RS-485. *In: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2016-03-29]. Dostupné z: http://cs.wikipedia.org/wiki/RS-485.

[34] Mono (software). *In: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2016-03-29]. Dostupné z: https://en.wikipedia.org/wiki/Mono_(software).

[35] C Sharp. *In: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2016-03-29]. Dostupné z: https://cs.wikipedia.org/wiki/C_Sharp.

[36] CALETKA, Ondřej. Let's Encrypt funguje v XP: věci se ale mohou rozbít. In: *Root.cz* [online]. 2016 [cit. 2016-03-31]. Dostupné z: http://www.root.cz/clanky/let-s-encrypt-funguje-v-xp-veci-se-ale-mohou-rozbit/

[37] Porting Winforms Applications | Mono. *Mono* [online]. [cit. 2016-03-31]. Dostupné z: http://www.mono-project.com/docs/gui/winforms/porting-winforms-applications/

[38] POSPÍŠIL, Miroslav a LUDVÍK, Vladimír. *TERMINOLOGIE Z OBLASTI METROLOGIE*. 2. ÚNMZ, 2010.

[39] ČESKÝ NORMALIZAČNÍ INSTITUT. *TNI 01 0115 : Mezinárodní metrologický slovník - Základní a všeobecné pojmy a přidružené termíny (VIM)*. Český normalizační institut, 2009.

[40] CITY TECHNOLOGY. *4P-75M CiTipeL Product Data Sheet*. 7. 2015.

[41] CITY TECHNOLOGY. *4CM CiTiceL Product Data Sheet*. 2. 2013.

[42] *NET Framework 4.5 and Windows XP*. [online]. [cit. 2016-03-31]. Dostupné z: https://blogs.msdn.microsoft.com/dotnet/p/dotnet45xp/

A Senzory plynů

K pochopení problematiky detektorů, legislativních opatření a principu kalibrace jsem se musel seznámit také se samotnými senzory plynů, jejich principem a měřícím řetězcem. Podle typu výstupu ze senzoru, se totiž rozlišuje postup prvotní kalibrace přístroje. Tato kapitola stručně seznamuje s problematikou detekce plynů, jednotlivých typů senzorů se kterými jsem se při práci setkal.

Detektory plynů jsou zařízení sloužící k detekci hladiny plynů nebo par v daném prostředí pomocí senzoru (čidla) plynu. Detekce je proces měření koncentrace plynu, pro kterou je detektor vyroben a kalibrován.

V počátcích detekce plynů využíval člověk výhod zvířat (jejich vyšší citlivosti oproti člověku) nebo základních fyzikálních principů. Příkladem těchto prvotních "detektorů"plynů, které nebyly velice přesné, jsou psi, kteří byli využívání pro detekci úniku metanu, kanárci pro výskyt kysličníku uhelnatého (CO), ale také například trvalý plamínek ohně indikující dostatek kyslíku v prostředí. Díky rozmachu rozvoje energetického, těžkého a chemického průmyslu došlo také k výraznému vývoji v oblasti detekce plynů. První detektory plynů se objevily v hornictví při těžbě a zpracování uhlí, kde se detekuje výskyt hlavně metanu (CH_4) a kysličníku uhelnatého (CO). Detekce plynů se v pozdější době rozšiřuje i na těžbu, dopravu a zpracování ropy a zemního plynu. Rozvoj v tomto odvětví byl zaznamenán také během první světové války, kde byly poprvé ve velké míře nasazeny chemické zbraně. Během výroby, přepravy a zpracování produktů v mnoha odvětvích průmyslu dochází ke vzniku toxických a hořlavých plynů. Tyto plyny jsou pro člověka jedovaté a také mohou výbuchem způsobit škody. Nebezpečné nejsou pouze plyny toxické a hořlavé, ale nebezpečné jsou také plyny inertní, které dokážou svou hmotností vytlačit kyslík z prostoru. Detekci plynů můžeme rozdělit podle účinků plynů na člověka a prostředí takto: [6]

- detekce hořlavých plynů, př: zemní plyn, amoniak, n-oktan,
- detekce toxických plynů, př: oxidy síry, oxidy dusíku, radon,
- detekce ostatních plynů, př: kyslík, dusík, argon.

Toxické plyny mohou být zároveň také hořlavé. Plyny poté tvoří v kombinaci se vzduchem výbušnou směs, která po překročení hraniční teploty hoří, výbuch lze považovat za speciální druh hoření plynu. Mezi tyto plyny se řadí například kysličník uhelnatý (CO) nebo čpavek (NH_3). V nižších koncentracích se plyny chovají jako toxické, kdežto ve vyšších koncentracích tvoří výbušnou směs a stávají se hořlavými. Spodní mez výbušnosti (anglicky LEL) je hraniční hodnota koncentrace plynu ve vzduchu, kdy se směs stane hořlavou. Pod hranici LEL směs obsahuje nedostatek plynu a přebytek oxidačního prostředku. Horní mez výbušnosti (UEL) je hraniční hodnota koncentrace plynu ve vzduchu, kdy směs přestane být hořlavou. Nad hranici UEL směs obsahuje přebytek plynu a nedostatek oxidačního prostředku. Pro lepší představu je

na obrázku 28 (obrázek převzat z článku [13]) vyobrazena grafika mezí výbušnosti. Aby došlo k samotnému výbuchu směsi, musí být splněn takzvaný výbuchový trojúhelník 27 (obrázek převzat z článku [9]) skládající se z: [4, 6]

- výbušné směsi,
- oxidačního prostředku,
- inicializačního zdroje.

Obrázek 27: Výbuchový trojúhelník.

Výbušná směs se vznítí pokud dojde ke splnění všech tří prvků z výbuchového trojúhelníku. Jako inicializační zdroje uvádí norma ČSN EN 1127-1 například: [4]

- statická elektřina,
- mechanické jiskry,
- elektrická zařízení,
- horké povrchy,
- plameny a horké plyny.

Obrázek 28: Vizualizace mezí výbušnosti.

Spodní mez výbušnosti je velice důležitá při procesu detekce plynů v prostředí pro prevenci proti výbuchu. Senzory určené pro tento druh detekce mají měřící rozsah 0 - 100% LEL, jelikož

úrovně koncentrace plynů, kdy je požadována signalizace nebo zásah pracovníků, jsou v normách a předpisech uváděny také v procentech LEL [6].

Pro detekci plynů existuje celá řada senzorů pracujících na různých principech, mezi nejčastěji používané typy senzory plynů patří:

- elektrochemické,
- polovodičové,
- infračervené,
- katalytické.

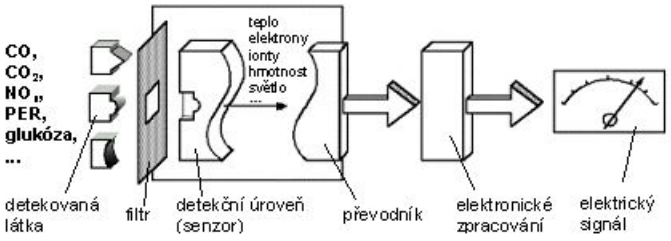
Výběr senzoru a jeho principu měření pro danou aplikaci se řídí mnoha aspekty, měřeným plynem, znečištěním okolního prostředí, interferujícími plyny, teplotou, vlhkostí, a tak dále. Podrobnosti o principech měření a výběru senzorů podle požadavků měření lze nalézt v ČSN EN 45544-4 a ČSN EN 60079-29-2 případně v předpise TPG 93801, ale tento předpis ve většině případů pouze přejímá údaje z předchozích uvedených norem.

O principech procesu detekce různých druhů senzorů plynů, které jsou nejčastěji používány a které mají vztah k programu, pojednávají následující podkapitoly, každá podkapitola je zaměřena na jeden konkrétní typ.

A.1 Elektrochemické senzory

Elektrochemické senzory mají založený princip detekce na změnách vodivosti elektrolytu, který se nachází mezi elektrodami. Obecný princip detekce elektrochemického senzoru je znázorněn na obrázku 29 (obrázek převzat z článku [14]). Způsobem jakým je elektrolyt vyroben, můžeme rozdělit tento druh detektorů na dvě skupiny: [15]

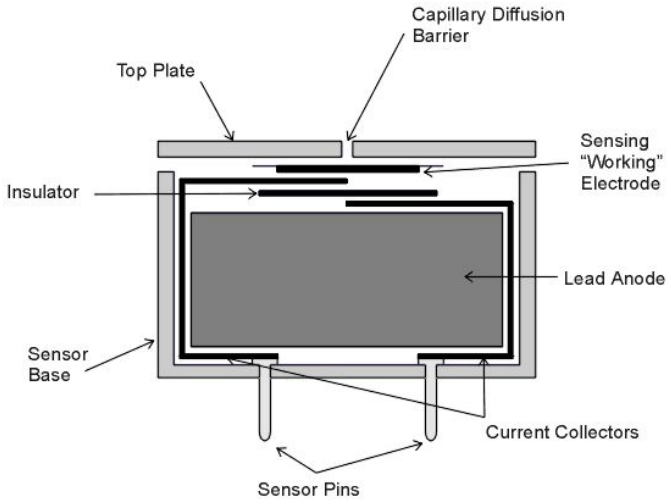
- s pevným elektrolytem,
- s tekutým elektrolytem (tlustovrstvé senzory).



Obrázek 29: Princip elektrochemického senzoru.

V případě typu pevného elektrolytu se nejčastěji používá elektrolyt na bázi ZrO_2 a u tekutých elektrolytů SnO_2. Senzor je složen ze systému dvou až čtyř elektrod v elektrolytu. Systém

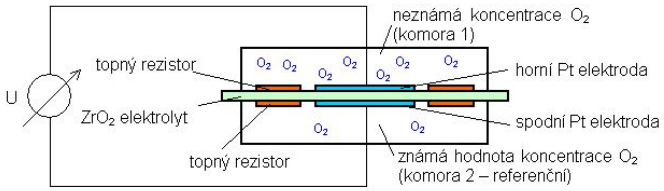
elektrod je oddělen od okolí membránou, která je propustná pouze pro molekuly analyzovaného plynu a nikoliv pro elektrolyt, vodu a ionty. Konstrukce je vyobrazena na obrázku 30 (obrázek převzat z článku [18]). Elektrolyt a molekuly měřeného plynu spolu reagují a na elektrodách dochází k oxidační a redukční reakci. Tyto reakce vyvolají změnu potenciálu galvanického článku, pokud vzroste koncentrace plynu, vzroste také potenciál mezi elektrodami článku. Elektrochemické senzory se většinou využívají v kombinaci s katalytickými senzory v přenosných detektorech plynů. Díky jejich relativně malé velikosti se obvykle umísťují ve větším počtu do jednoho zařízení. Využití tohoto druhu senzoru je při měření koncentrace kyslíku, toxických plynů a především instalace v přenosných detektorech. Výhodou je nízká cena a pro běžné plyny je senzor spolehlivý. Mezi nevýhody patří dlouhá odezva detekce, možnost poškození vysokou koncentrací plynu, vysoká cena pro speciální plyny [7, 10, 11, 14, 15, 18, 19].



Obrázek 30: Princip elektrochemického senzoru O_2.

Senzory s pevným elektrolytem: neobsahují kapalný elektrolyt a tudíž odpadá nutnost periodického vyměňování elektrolytu. Díky tomu je konstrukce senzoru menší a provozuschopnost bez zásahu vyšší. Obrázek 31 (obrázek převzat z článku [15]) znázorňuje kyslíkový senzor k detekci koncentrace kyslíku ve spalinách motorových vozidel. Senzor je vyhříván na teploty nad 300 °C, jelikož pod touto teplotou je pohyblivost kyslíkových iontů v elektrolytu velmi malá a elektrolyt není vodivý. Běžně se senzor vyhřívá na teploty 600 - 800 °C. Jedna elektroda je v kontaktu s analyzovaným plynem a druhá je vystavena referenční hodnotě koncentrace kyslíku, ve většině případů se jedná o vzduch. Příkladem senzoru s pevným elektrolytem je lambda sonda [15].

Senzory s tekutým elektrolytem: hlavní nevýhodou těchto senzorů je samotný tekutý elektrolyt, který ovlivňuje životnost (1 až 3 roky) senzoru svým vypařováním. Elektrolyt se musí pravidelně obměňovat, nebo se provádí výměna celého senzoru. V současné době se vyrábějí většinou senzory bez možnosti výměny, nebo doplňování elektrolytu. Membrána oddělující elektrolyt od okolí zpomaluje proces vysychání, ale zároveň snižuje citlivost senzoru. Princip procesu detekce je popsán na začátku této podkapitoly a konstrukce senzoru je vyobrazena na ilustraci 30. Nevý-



Obrázek 31: Elektrochemický senzor s pevným elektrolytem.

hoda v životnosti senzoru je vykompenzována klady této konstrukce, které jsou dobrá citlivost, selektivita a široké spektrum detekovatelných plynů [15].

A.2 Měřící řetězec elektrochemického senzoru

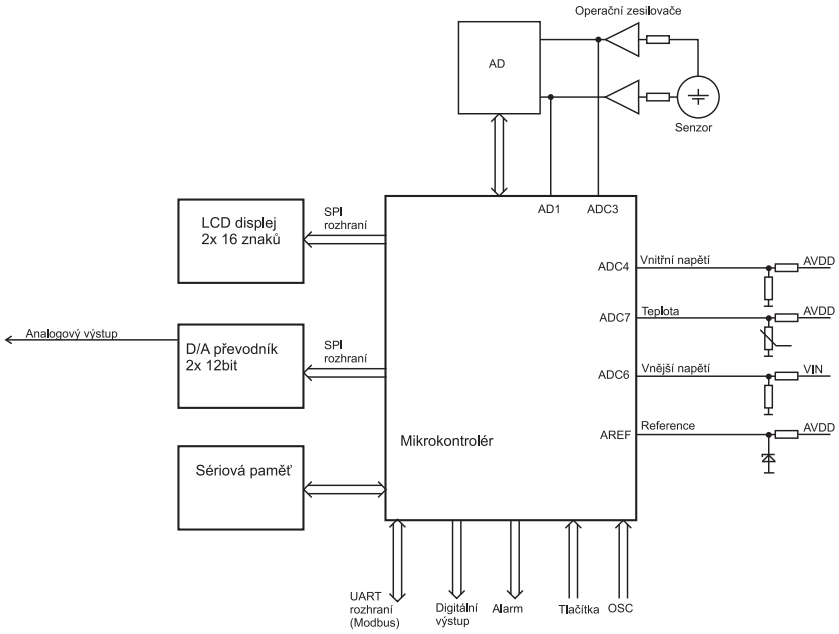
Konstrukce senzoru poskytuje signál ve formě elektrického proudu. Úrovně tohoto proudu jsou řádově desítky nanoampér na jeden ppm. Například senzor 4CM od firmy CityTechnology má citlivost 70 nA/ppm, takže při 30 ppm CO, což je maximální povolená trvalá hranice CO v ovzduší, dává na svém výstupu 210 nA. Tento proud je nutno zesílit na úrovně vhodné pro měření, tudíž je nezbytné použít vysoce kvalitní operační zesilovače, neboť běžné operační zesilovače mají vlastní chyby pod měřenými úrovněmi. Velkou výhodou tohoto principu je linearita výstupního signálu v závislosti na měřené koncentraci [41].

Měřící a indikační řetězec je složen ze senzoru, zesilovače, AD převodníku, výpočetní jednotky, zobrazovače, DA převodníků a výstupních zesilovačů, obrázek 32. Výstupem z měřícího řetězce je bezrozměrná hodnota N z AD převodníku, která je přímo úměrná napětí vystupujícímu ze zesilovače, a toto napětí je přímo úměrné měřené koncentraci. Pomocí lineární rovnice je N přepočítáno na koncentraci. V rámci indikačního řetězce je koncentrace přepočítána v požadovaných mezích na analogovou hodnotu, frekvenci a nebo předávána jako číselná hodnota přes protokol MODBUS.

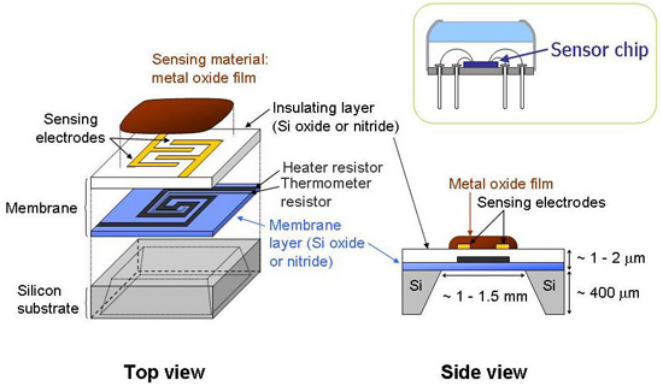
A.3 Polovodičové senzory

Detekční princip je založen na změnách vodivosti citlivé polovodičové vrstvy působením analyzovaného plynu, základním principem jsou tedy procesy sorpce plynů. Nejčastěji se využívá proces chemisorpce, během které vzniká mezi molekulami plynu a molekulami citlivé polovodičové vrstvy chemická vazba a přenos náboje. Jelikož proces chemisorpce (vytvoření chemické vazby) vyžaduje aktivační energii, tak se polovodičová vrstva a křemíková destička vyhřívají na 200 - 400 °C. Citlivá polovodičová vrstva, která je nanesena na křemíkovou destičku, může být vyrobena z různých oxidů kovů (například SnO_2, ZnO_2 a dalších), nejčastěji se využívá oxid číničitý (SnO_2), jelikož vyniká dobrou přesností a nízkou pracovní teplotou (od 200 °C). Stavba senzoru je znázorněna na obrázku 33 (obrázek převzat z článku [16]) [6, 7].

Pokud se citlivá vrstva vyskytne v kontaktu s oxidačním plynem, tak dojde k pokrytí citlivé vrstvy kyslíkem, který odebírá z vrstvy část elektronů. Na povrchu tedy vznikne ochuzená



Obrázek 32: Měřící řetězec elektrochemického senzoru.

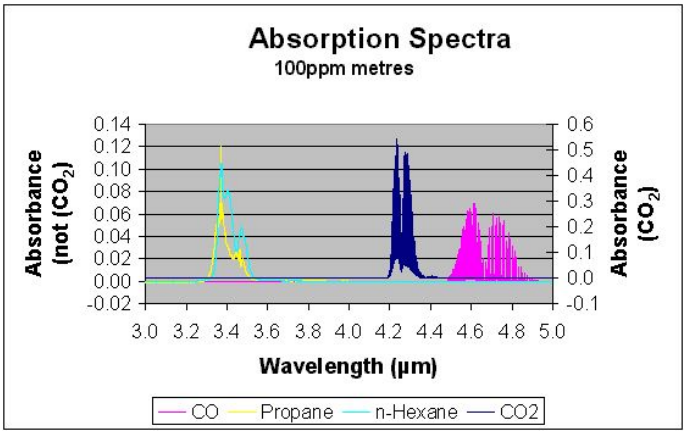


Obrázek 33: Polovodičový senzor.

oblast s ionty kyslíku a vodivost vrstvy se výrazně klesne. V případě kontaktu s redukčním plynem, dochází k reakci molekul plynu a molekul kyslíků na citlivé vrstvě a dojde ke zvýšení vodivosti vrstvy. Tyto změny vodivosti se poté zpracovávají (zesilují, komparují). Předností této konstrukce je nízké náklady, rychlá odezva, relativně velká citlivost. Nevýhody jsou velká nepřesnost, malá stabilita parametrů, závislost na vlhkosti a teplotě okolí, nelinearita výstupního signálů. Životnost je vysoká, až 10 let, protože nedochází ke spotřebovávání snímacího materiálu. Vzhledem ke kombinaci výhod a nevýhod se tento typ senzorů využívá v jednoduchých detektorech pro nenáročné provozy a také v přenosných detektorech úniku plynárenských zařízení a rozvodů, kde není důležité stanovit přesnou koncentraci plynu, ale detekovat jeho únik [6, 7].

A.4 Infračervené senzory

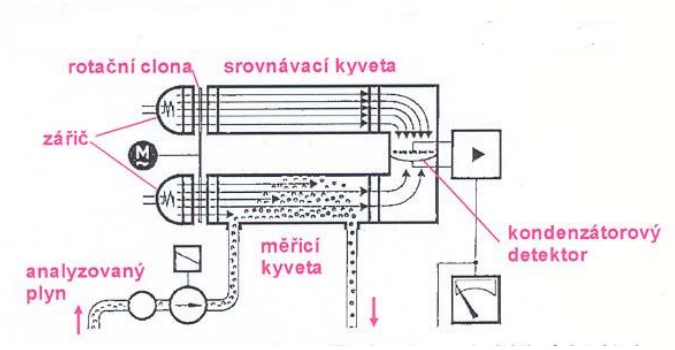
Lambert-Beerův zákon pojednává o matematické závislosti absorpce monochromatického elektromagnetického záření od vlastností materiálu, přes který prochází záření. Tohoto zákona je využito u infračervených senzorů, kde tímto zákonem víme, že ve spektru infračerveného záření existují vlnové délky záření, které jsou absorbovány určitými plyny, jejichž molekula je složena z alespoň dvou různých atomů. Souměrné molekuly (například N_2, 0_2, H_2) totiž infračervené záření neabsorbují. Obrázek34 (obrázek převzat z článku [21]) znázorňuje absorpce vlnových délek různými plyny. Nejčastěji se při analýze používá záření vlnových délek od 0.7 µm do 10 µm. Konstrukce senzoru je vyobrazena ilustrací 35 (obrázek převzat z článku [10]). Ze zdrojů infračerveného záření vychází záření, které jsou synchronně přerušované rotační clonou a prochází komorami (kyvetami) mezi sebou oddělenými kovovou membránou tvořící jednu elektrodu kapacitního snímače tlaku. Záření procházející měřící komorou se částečně absorbuje (intenzita paprsku se zmenší), pokud se v měřící komoře vyskytuje analyzovaný plyn. Ve srovnávací (referenční) komoře se vyskytuje plyn o známé koncentraci. Pomocí srovnávacího záření se kompenzuje vliv teploty, tlaku a dalších vnějších vlivů. Při absorpci infračerveného záření dochází k ohřevu obsahu komory a změnám tlaků, ty jsou měřeny membránou kapacitního snímače tlaku, na který dopadají záření z obou komor. Selektivita se dosahuje nejčastěji použitím selektivního detektoru (měřený plyn ve srovnávací komoře). Výstupní signál je střídavý a podle koncentrace se mění amplituda [6, 10, 12, 21].



Obrázek 34: Absorpce záření pro různé plyny.

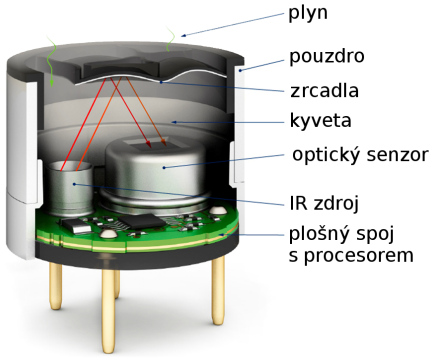
Konstrukci těchto čidel lze rozdělit na dva druhy, a to bodová konstrukce a konstrukce v otevřeném prostoru. Bodová konstrukce je na jednom místě to znamená, že vzdálenost, kterou musí infračervené záření urazit v komorách od zdroje ke snímači je v řádech několika centimetrů až milimetrů. Konstrukce v otevřeném prostoru spočívá ve dvojici vysílačů a přijímačů umístěných až 200 m od sebe. Jeden paprsek slouží jako měřící a má vlnovou délku, kterou pohlcuje měřený plyn. Druhý paprsek je referenční, pomocí kterého se určuje zda se nesnížila intenzita měřícího paprsku z jiných důvodů (například mlha, déšť a další) než detekce daného plynu. Dojde-li k za-

znamenání poklesu intenzity měřícího i referenčního paprsku, tak není tento pokles vyhodnocen jako detekování plynu. Pokud ovšem dojde k poklesu intenzity pouze u měřícího paprsku, je pokles vyhodnocen jako detekování plynu. Příklady obou druhů konstrukce jsou ukázány na obrázcích 36 (obrázek převzat z článku [12]) a 37 (obrázek převzat z článku [17]) [6].

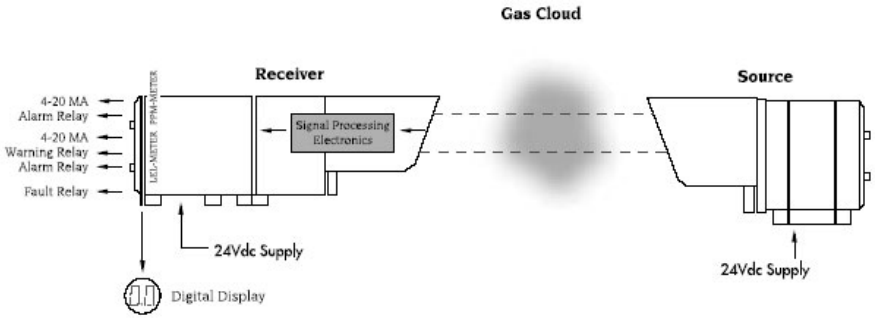


Obrázek 35: Princip infračerveného senzoru.

Infračervené senzory jsou náchylné na vodní páry a některé druhy látek. Mezi jejich výhody ale patří rychlost vyhodnocení, dlouhá životnost, měření v inertním prostředí, velká selektivita. Nevýhodou je vysoká cena a složitá kalibrace, jelikož výstup senzoru není lineární [6].



Obrázek 36: Bodová konstrukce infračerveného senzoru.



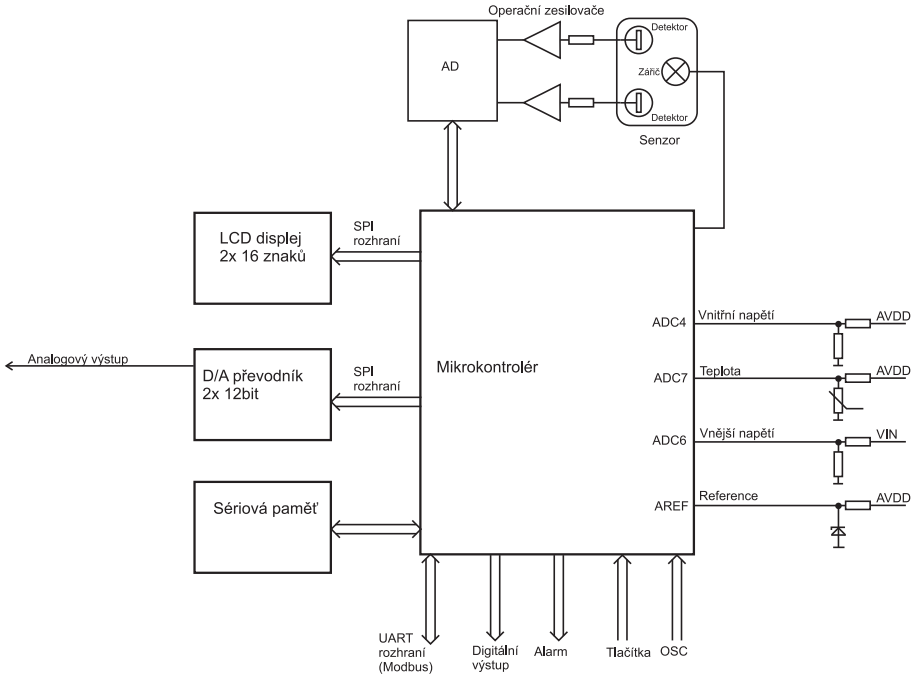
Obrázek 37: Konstrukce infračerveného senzoru v otevřeném prostoru.

A.5 Měřící řetězec infračerveného senzoru

Výstupní signál ze senzorů pracujících na infračerveném principu je velmi komplexní záležitosti. Vysílané infračervené záření je rytmicky přerušované, toto přerušování se přenáší na signál detektorů. Měřící detektor poskytuje signál o vrcholové hodnotě cca 200 mV, změna přes celý měřící rozsah koncentrace způsobí změnu signálu o cca 2 mV proti referenčnímu detektoru.

Odečtením hodnot z referenčního a měřícího detektoru se získává signál pro další zpracování. Změna teploty a koncentrace nejen ovlivňuje amplitudu rozdílového signálu, ale různě i amplitudu signálu z jednotlivých detektorů. Z tohoto je zřejmé, že měřící řetězec není jednoduchý a vyžaduje použití kvalitních komponent, je nejnáročnější ze všech principů.

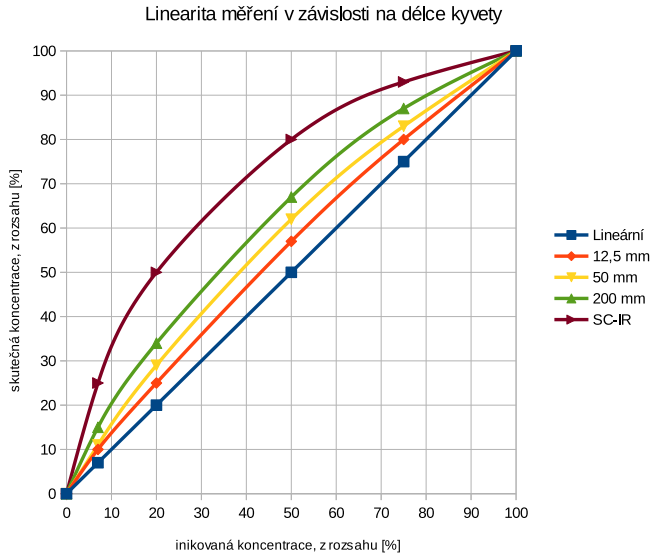
Měřící a indikační řetězec je složen z budiče infračerveného signálu, senzoru, zesilovače referenčního signálu, zesilovače měřeného signálu, AD převodníků jednotlivých signálů, výpočetní jednotky, zobrazovače, DA převodníků a výstupních zesilovačů, vyobrazeno obrázkem 38. Výstupem z měřícího řetězce jsou bezrozměrné hodnoty AD převodníků. Tyto hodnoty spolu s několika kalibračními konstantami jsou dále zpracovány matematickým aparátem a je vypočtena koncentrace. V rámci indikačního řetězce je koncentrace přepočítána v požadovaných mezích na analogovou hodnotu, frekvenci a nebo předávána jako číselná hodnota přes protokol MODBUS. Výpočet kalibračních konstant a jejich předání snímačům je jedním z úkolu realizovaného programu.



Obrázek 38: Měřící řetězec infračerveného senzoru.

Závislost mezi hodnotami vyčítanými z AD převodníku a koncentrací je exponenciální. Volbou vhodné délky kyvety, část senzoru přes který prochází infračervené záření a zároveň měřený

plyn, lze měření umístit do počáteční části exponenciály a tím minimalizovat chybu způsobenou nelineární závislosti. Tuto optimalizaci kyvety si můžeme dovolit u laboratorních přístrojů, kde nejsme omezování rozměry kyvety přístroje. U přístrojů do průmyslového provozu je většinou rozměr senzoru limitující, kde je již dán vnější rozměr nějakým standardem a zároveň je snaha po co největší citlivosti. Proto tyto senzory využívají i horní část exponenciály. Závislost linearity měření na délce kyvety je zobrazena grafem na obrázku 39, závislost absorpce na délce kyvety a koncentraci je na obrázku 40.



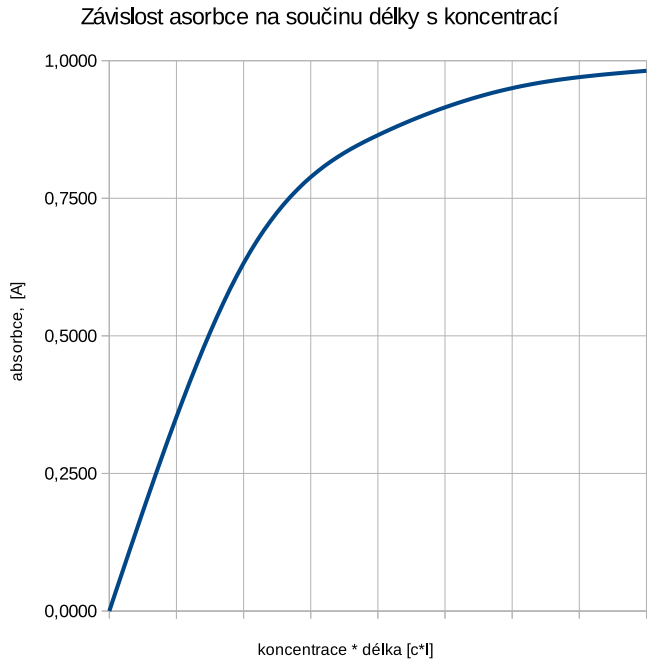
Obrázek 39: Linearita měření v závislosti na délce kyvety.

Samozřejmě tato nelinearita nevadí u přístrojů, které jsou označovány jako detektory, tj. nemají žádný indikační výstup, na kterém by bylo možno odečíst okamžitou hodnotu, ale pouze indikují dosažení určité hranice koncentrace například pomocí relé. V minulosti, kdy indikační přístroje byly analogové se cejchování provádělo přímo označením příslušné koncentrace na stupnice přístroje nebo zapisovače. V současné době, kdy je analogová hodnota převedena na digitální, se číselné vyjádření linearizace provádí výpočtem pomocí linearizační rovnice. Výpočet provádí vnitřní mikroprocesor, který následně nastavuje indikační výstupy.

Výpočet a zadání linearizačních konstant se provádí pouze při prvotní kalibraci nebo při výměně senzoru, při dalších kalibracích stačí provádět pouze lineární kalibraci s jedním plynem.

A.6 Katalytické senzory

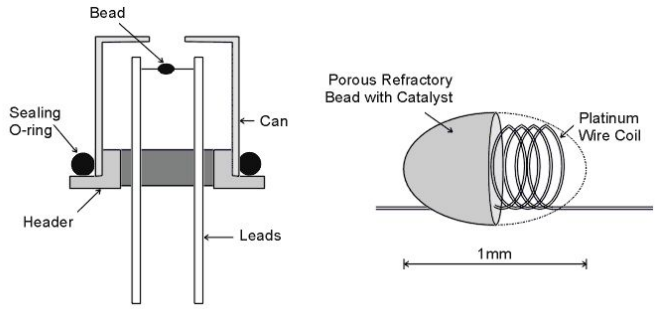
Pro měření koncentrace hořlavých plynů a par lze využívat princip měření tepelného zabarvení, které se vyskytuje ve spalovací reakci hořlavých látek. Tento spalovací proces se děje na senzorech s katalyticky účinným povrchem, katalytické senzory známe také jako pelistory. Senzor je složen z měřící komory s elektricky žhaveným topným tělískem, které má katalyticky účinný povrch,



Obrázek 40: Závislost absorpce na délce kyvety a koncentraci.

kde dochází ke katalytickému spalování detekovaného plynu. Teplem uvolněným při katalytickém spalování se zvyšuje teplota tělíska a tím zároveň dojde ke změně rezistivity tělíska. Pelistor je vlastně typ kalorimetrického senzoru, který vyhodnocuje koncentraci plynu na základě množství uvolněného tepla během katalytického spalování. Pro výrobu tělíska pelistoru je nejvhodnější materiál platinový drátek o průměru 0,03 - 0,1 mm, jelikož platinový drátek umožňuje, aby splňoval funkci nosného elementu pro drátek a pouzdro senzoru, odporového teploměru a také jako topné tělísko pro proces katalyzace. Topné tělísko vyhřívá pelistor na pracovní teplotu mezi 500 - 600 °C. Katalytický senzor se nejčastěji vyrábí ve formě keramického válce nebo perličky o průměru 1 - 2 mm na bázi oxidu hlinitého. Katalytický povrch bývá vyroben z platiny nebo případně ze směsi platiny a paládia. Katalytický senzor (pelistor) je znázorněn na obrázku 41 (obrázek převzat z článku [20]) [6, 8, 10, 20].

Pro získání výstupního signálu senzoru se pelistor zapojuje do Wheatstoneova můstku zároveň s druhým referenční pelistorem, vyrobeného ze stejných materiálů, který ale nemá katalytický povrch. Podobně jako u ostatních typů senzorů, i zde funguje referenční pelistor jako kompenzace vnějších rušivých vlivů (změny teplot, tepelné vodivosti či průtoku plynu) ovlivňujících výstupní signál. Oba pelistory, jak měřící tak referenční, se mohou umísťovat do oddělených komor nebo do jedné společné komory. Membrána oddělující pelistory od okolí bývá vyrobena z kovového porézního materiálu nejčastěji složeného z sintrovaného bronzu nebo niklu. Porézní membrána umožňuje přístup plynu do měřící komory a zároveň zabraňuje v případě vznícení výbušné směsi uvnitř komory k průniku plamene mimo měřící komoru do vnějšího okolí. Výstup



Obrázek 41: Katalytický senzor.

senzoru je lineární a měřící rozsah se udává v 0 až 100 % LEL. Výhodou je jednoduché provedení, vysoká životnost (5 a více let), provozní spolehlivost, jednoduchá údržba. Nevýhodou je negativní vliv katalytických jedů na aktivitu katalyzátoru. Při měření se využívají dva pracovní režimy pelistorů: [6, 8, 11, 20]

- neizotermní,
- izotermní.

Neizotermní režim je založen na změnách teploty na měřícím tělísku a jeho rezistivity při katalytickém spalování. Změna rezistivity se projeví změnou rovnováhy v Wheatstoneově můstku a změnou výstupního signálu, který je úměrný ke koncentraci plynu. Vyhodnocování výstupního signálu lze provádět dvěma způsoby, a to použitím můstku s konstantním napájecím napětím, nebo můstkem s konstantním napájecím proudem. Konstantní napětí se využívá u přenosných detektorů z důvodu menší spotřeby. Zapojení obvodu je v neizotermním případě jednoduché a využívá se nejčastěji v komerčních detektorech plynů.

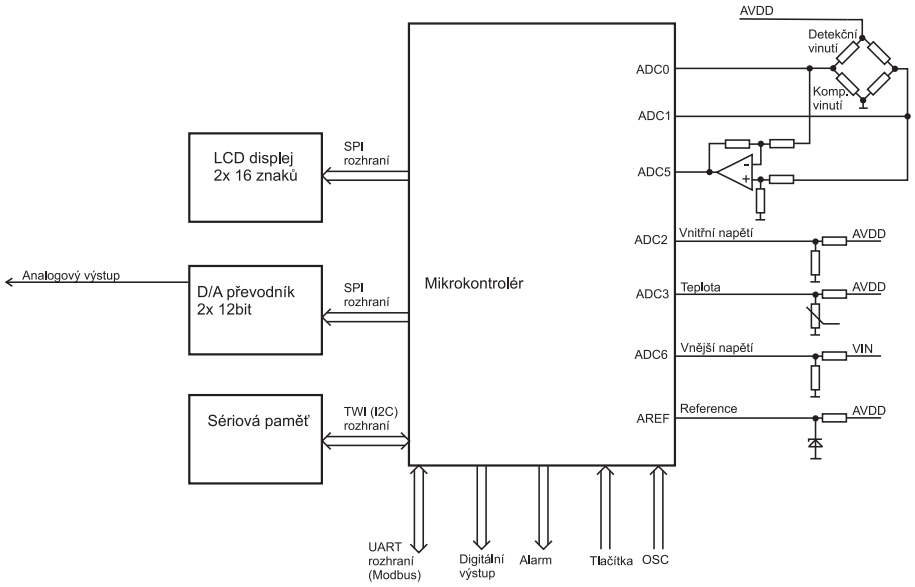
U izotermního režimu se teplota měřícího tělíska udržuje stále konstantní pomocí regulace napájení tělíska. Koncentrace plynu se odvozuje od hodnoty spotřeby elektrické energie nutné k udržení konstantní teploty měřícího tělíska při katalytickém spalování, elektrický příkon je úměrný koncentraci plynu. Metoda se využívá pro dosažení větší selektivity hořlavých plynů v několika složkových směsích.

A.7 Měřící řetězec katalytického senzoru

U katalytického senzoru, zapojeného dle doporučení výrobce, poskytuje měřící můstek napětí, které se zesiluje operačními zesilovači. Zde již nejsou, tak extrémní požadavky na kvalitu používaných operačních zesilovačů. Používané senzory mají citlivost řádově desítky milivoltů na procento objemové koncentrace. Například senzor 4P-75M od firmy CityTechnology má citlivost 24 mV/% [40].

Velkou výhodou tohoto principu, stejně jako u elektrochemického, je linearita výstupního signálu v závislosti na měřené koncentraci.

Měřící a indikační řetězec je složen ze senzoru, zesilovače, AD převodníku, výpočetní jednotky, zobrazovače, DA převodníků a výstupních zesilovačů. Řetězec je znázorněn na obrázku 42. Výstupem z měřícího řetězce je bezrozměrná hodnota N z AD převodníku, která je přímo úměrná napětí vystupujícímu ze zesilovače, a toto napětí je přímo úměrné měřené koncentraci. Pomocí lineární rovnice je N přepočítáno na koncentraci. V rámci indikačního řetězce je koncentrace přepočítána v požadovaných mezích na analogovou hodnotu, frekvenci a nebo předávána jako číselná hodnota přes protokol MODBUS.



Obrázek 42: Měřící řetězec katalytického senzoru.

B Přílohy na CD-ROM

- Příloha 1 – Text diplomové práce ve formátu PDF,
- Příloha 2 – Zdrojové kódy projektu programového vybavení (Microsoft Visual C# 2010 Express).